第一章　PHP & MySQL 網頁製作

如果你選擇了 Linux 作爲你的伺服器，那使用 PHP & MySQL 製作網頁資料庫是最自然不過的。這樣的組合，就像是你使用 NT 作爲伺服器，那你就會選擇 ASP+SQL 製作網頁資料庫一樣，當然，天空是無限寬廣的，明天也許會有更好的選擇，但今天，這樣的選擇應該是最自然的。

第一項　測試 PHP & Mysql

apache 的設定我們在 Linux 中已經介紹過了，這裡不再重述。
要測試 PHP 是否已經提供服務，只需在網頁的目錄中，建立一個副檔名爲.php 的檔案，再透過瀏覽器瀏覽該網頁是否能夠正確的顯示。

測試 PHP

請依底下步驟新增檔案：

```
1. cd /var/www/html
2. vi phpinfo.php
3. 輸入內容後儲存離開，如下：
   <? phpinfo(); ?>                    ‘呼叫 phpinfo()函數
4. 使用瀏覽器瀏覽該網頁，例如：
   http://localhost/phpinfo.php
```

如果可以看到有顯示 PHP 的相關訊息，就表示它已經啓動服務了。
另，由於 RedHat 7.0 之前是使用 PHP 3 的套件，所以網頁的副檔名存成『.php3』，而 RedHat 7.0 之後使用的版本是 PHP 4，所以網頁的副檔名存成『.php』。

測試 MySQL

測試 MySQL 是否啓動，直接輸入：

```
mysqlshow
```

如果有看到 Database 的表格出現，就表示 MySQL 已經啓動，但是，還沒有設定使用者的帳號、密碼，所以任何使用者都可以刪除其中的重要資料，因此，必須先執行類似『mysqladmin -u 使用者 password ‘密碼’ 』的動作來新增系統管理員的帳號、密碼：

```
mysqladmin -u root password ‘1234’
```

設定之後，若需要檢視資料庫的話，就必須加上帳號及密碼囉！例如：

```
mysqlshow -p                              ‘這種方式的安全性較高
1234
```

或者是

```
mysqlshow -p1234                          ‘我使用 root 登入，因此省略帳號
```

第二節　CGI & PHP

第一項　cgi

在網頁上要使用表單，通常是在 server 端要有 cgi（Common Gateway Interface）程式負責接受資料、處理資料。通常撰寫 cgi 的程式會使用 C、perl，我們可以將它當做是 dos 下的批次檔、Linux 下的 shell script 檔案，都是在主機上執行的程式。
cgi 不能算是程式語言，它只是 interface，是使用其它的程式語式所撰寫的程式，我們現在所學習使用的 PHP，就可以使用 PHP 來撰寫 cgi 的程式。

第二項　PHP

使用 PHP 撰寫 cgi 程式有幾個優點：

- 簡單：撰寫程式比較簡單，處理資料的方式也簡單；原來使用 perl 撰寫的程式在處理特殊符號、中文時，實在很麻煩，而 PHP 幫我們解析、簡化這些動作了。
- 支援豐富：PHP 提供了大量的函數，可以支援使用的產品有很多，例如資料庫、電子郵件…都可以透過 PHP 使用。
- 效能：使用 PHP 處理資料的效能要好的多。

第二章　簡簡單單的學 PHP

PHP 的程式是內崁式的程式語言，和我們一般知道的 VBScript、JavaScript 類似，都是直接加在 HTML 的網頁中，不同的是，VBScript、JavaScript 都是在 Client 端執行的描述語言，而 PHP 則是在 Server 端執行的描述語言；前者是連著 HTML 網頁一起交給瀏覽器解讀，後者是在 Server 由 PHP 解讀後送至 Apache，再經由 Apache 將 HTML 傳回到瀏覽器，它是一個單純的 HTML 文件，另外，由於 PHP 是在 Server 處理，所以對 Server 的負載會比較大。

第一節　第一個 PHP 程式

PHP 和 HTML 有些類似，學過 HTML 後再來學 PHP 一定會覺得很容易的。PHP 的程式必須放在『<?…?>或<?php…?>』的標籤之中。舉個例子：

```
'檔案名稱記得要存成『.php』或『.php3』
<HTML>
<BODY>
    <p>現在的日期時間是：                    '一般的 HTML 標籤
    <?                                       'PHP 啓始
        echo date("Y-m-d H:i:s");
    ?>                                       'PHP 結束
    </p>
</BODY>
</HTML>
```

第一項　結束符號

在 PHP 之中，每一行敘述的結束，都必須加上『;』，做爲該行敘述的結束符號。如果沒有加上結束符號，會使 PHP 產生錯誤訊息，而且，會顯示是第幾行產生的錯誤訊息。同時，請你檢查看看它的上一行是不是因爲忘了加上『;』符號所引起的錯誤。

第二項　echo

由於在<?…?>標籤中的內容實際上都是由 PHP 程式來負責處理，而不是 Apache，因此，在 PHP 中運算、處理的結果都只是存放在 PHP 之中，我們必須使用 echo 敘述，將內容傳回去給 Apache，再由 Apache 將 HTML 文件傳回給使用者的瀏覽器。

所以，echo 是將資料輸出到 HTML 文件的敘述，這樣子，訪客才可以看到 PHP

所產生的網頁。echo 除了輸出文字內容之外，還可以同時輸出標籤，例如：

```
echo “<p>輸出標籤及內文</p>”;
```

第三項　date()

這是 PHP 所提供的一個函數，主要的功能在傳回系統的日期、時間。
函數，就像數學上的函數類似，例如：

```
f(x)=x²+2x+1
```

其中的『f(x)』就是函數，我們只需要指定其中的 x 是多少，自然就可以得到『x^2+2x+1』的值。
電腦中的函數也是一樣，給它一個值（或多個、也有些函數不需要），函數會將計算的結果傳回來給我們。
在『echo date("Y-m-d H:i:s");』敘述中，我們給 date( )函數的值是『Y-m-d H:i:s』，而 date( )函數在運算後將『2001-08-09 16:22:15』的結果傳回來，因此，這一行的敘述就會像是『echo “2001-08-09 16:22:15”;』囉。
使用函數的優點在於，函數每次執行時，可能因爲條件不同、或執行的時間不同，而會有不同的結果，這個也就是函數好用的地方。

第四項　使用 PHP 的時機

另外，PHP 的標籤不是只能有一組，只要有需要，我們可以在 HTML 文件中的任何位置使用<?…?>標籤，加上 PHP 的程式，例如我們將上面的例子修改如下：

```
<HTML>
<BODY>                                    'PHP 和 HTML 一樣，可以放在同一行
    <p>今天的日期是： <? echo date("Y-m-d"); ?> </p>
    <p>現在的時間是： <? echo date("H:i:s"); ?> </p>
</BODY>
</HTML>
```

我相信，再加上 CSS 樣式，也沒有問題的，例如：

```
<HTML>
<BODY>
    <p style="color:green;">今天的日期是： <? echo date("Y-m-d"); ?> </p>
    <p style="color:blue;">現在的時間是： <? echo date("H:i:s"); ?> </p>
</BODY>
</HTML>
```

我們已經學過 CSS 了，因此，在往後的程式中，你隨時可以透過 CSS 將呈現出來的網頁內容，在樣式上做些變化。

第二節　變數

變數的作用可以是代表目前運算的結果、也可以是代表某一個字串、也可以記錄某些特定資料，總言之，就是爲了讓程式能更方便、更靈活，而使用變數來代表特定的『值』、或『資料』。
在 PHP 中，只要看到英文字前面有加上『$』的就是變數了，而 PHP 的變數使用起來很靈活，沒有太多的限制，就當成是不定型變數來使用就行了。
舉個例子來看看變數的使用：

```
<BODY>
    <?
        $x=date("Y-m-d");              ‘將 date() 傳回的資料指定給$x 變數
        $y=date("H:i:s");              ‘將 date() 傳回的資料指定給$y 變數
    ?>
    <p>今天的日期是： <? echo  $x; ?> </p>      ‘印出$x 的值
    <p>現在的時間是： <? echo  $y; ?> </p>      ‘印出$y 的值
</BODY>
```

在程式語言中使用『=』的方式和數學上的意義、功能不同，在這裡，使用『=』是將右邊的值指定給左邊的變數。底下的例子在數學上是不合理的，但是，在程式中我們常常會拿來運用：

```
$z=10;                                 ‘指定$z 等於 10
$z=$z + 1                              ‘右邊$z+1 的結果是 11，再指定給左邊的$z
$z=$z + 5                              ‘右邊$z+5 的結果是 16，再指定給左邊的$z
```

在 PHP 中的變數是有分大小寫的，這點請千萬注意！

第三節　遞送資料給 PHP

第一項　get

附加在網址後面傳送，簡單，但可傳送的資料較少。
我們做個實際的例子，就會明白的，如下：

```
‘存檔：t3.php3
<BODY>
    <?
        $sum=$ca+$cb+$cc;              ‘分別取得 &ca, &cb, &cc 三個變數的值
        $avg=$sum/3;
    ?>
    <p>國文分數是： <? echo  $ca; ?> </p>
    <p>英文分數是： <? echo  $cb; ?> </p>
```

```
    <p>數學分數是： <? echo  $cc; ?> </p>
    <p>三科總分是： <? echo  $sum; ?> </p>
    <p>平均分數是： <? echo  $avg; ?> </p>
</BODY>
```

至於『$ca, $cb, $cc』三個變數的值，是透過網址以這種形式傳送：

```
http://localhost/t3.php3?ca=100&cb=90&cc=80
```

使用『?』表示網址後面是附帶的資料，使用『ca=100』表示傳遞資料的名稱是 ca、傳遞的值等於 100，使用『&』來區隔不同的名稱、值。
傳遞到 PHP 的程式中，直接在名稱前面加上『$』，就可以當做變數來使用，因此我們可以直接使用『$ca, $cb, $cc』這三個變數。
要特別注意的是，由於在 Linux 系統中英文大小寫是有分別的，因此，在網址後面附加資料傳遞給 PHP 程式、PHP 程式將資料當做變數使用，二者間的英文名稱大小寫要相同。

第二項　post

使用表單方式傳送，可以傳送大量資料。
我們建立一個表單，和剛才建立的 PHP 網頁一起配合運作，表單內容如下：

```
'存檔：form1.htm
<HTML>
<BODY>
    <H1>成績輸入表單</H1>
    <FORM method=post action=t3.php3>
        <p>國文分數： <INPUT type=text name=ca> </p>
        <p>英文分數： <INPUT type=text name=cb> </p>
        <p>數學分數： <INPUT type=text name=cc> </p>
        <p><INPUT type=submit value="送出資料">
           <INPUT type=reset value="清除重來">
    </FORM>
</BODY>
</HTML>
```

按下『送出資料』後，注意看一下網址，會發現表單會將資料送至 action 指定的 PHP 網頁處理，再由 PHP 網頁將資料以 HTML 形式呈現出來。
這個例子中我們是以『method=post』方式來傳送資料，如果，我們在網頁中是使用『method=get』來傳送資料，結果也是一樣，只是，如果表單中有較大量的資料要傳送時，就一定要使用『method=post』的方式來傳送。

第三項　圖像式的傳送鈕

如果你覺得平常的傳送鈕太過單調，還可以使用影像來代替，使用的方式如下：

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

資料傳送到 action 所指定的網頁時，除了各個欄位的資料傳送過去之外，另外，還同時傳送『名稱_x』、『名稱_y』這二個變數，分別代表在圖像式的按鈕上點下的 X 軸、Y 軸的位置，例如，前面指定的 name 是 sub，因此，就有 sub_x, sub_y 這二個變數，如下：

```
<p>X： <? echo  $sub_x; ?> </p>        '印出 X 軸位置
<p>Y： <? echo  $sub_y; ?> </p>        '印出 Y 軸位置
```

要記得，這二個變數名稱是依『<input type=image src="image.gif" name="sub">』之中的『name』名稱而改變的。

第三章　PHP 程式設計基礎

第一節　資料與運算

第一項　字串符號

在『"』雙引號之中的內容會被當成是字串，但是 PHP 在遇到『$』符號時，還是會將變數換成變數所代表的值（變數前後要空格），這個功能在我們撰寫 PHP 網頁時很有幫助，可以使我們在製作 PHP 網頁更簡單、更清楚，就不必再使用太多的字串連結符號。看看底下的例子：

```
$a = "Hello ";
echo "<P> $a";                    ‘會輸出成 "<P> Hello "
```

另外有一個『'』單引號，也是字串符號，它比較強力，被單引號包圍的都是文字，不再具有其它意義，例如：

```
echo ‘<P> $a’;                    ‘會輸出成 ’<P> $a ‘
```

第二項　字串連接符號

要連接二個字串，可以使用字串連接符號『.』，將二個字串連接成一個字串，來看看一個簡單的例子：

```
$a = "Hello ";
$b = $a . "World!";               ‘$b="Hello World!"
echo $b;

$a = "Hello ";
$a .= "World!";                   ‘等於 $a = $a . “World!” = "Hello World!"
echo "<P>" . $a;                  ‘輸出 "<P>Hello World!"
```

再來看一個例子，這二行輸出的結果是相同的：

```
echo "<P>" . $a;
echo "<P> $a";                    ‘和上一行的功能相同，但沒有加『.』
```

至於在『"』雙引號之中的『.』就不是『字串連接符號』，而只是單純的句點了！

第三項　常數

可以在程式中隨時指定新值，我們稱爲變數，相反的，常數的值就是固定不變的，PHP 中已經存在幾個常數，例如：

常數	說明
__FILE__	代表目前瀏覽的檔案，以絕對位置的方式呈現。前後均有二個底線。
__LINE__	顯示這一行 script 由上算下來是第幾行。前後均有二個底線。
PHP_VERSION	顯示 PHP 執行的版本
PHP_OS	顯示作業系統的名稱

除此之外，我們還可以自己定義常數，定義常數的方式如下：

```
define("PI", 3.1415926);              ‘定義常數 PI=3.1415926
define("HA", "Hello World !!");       ‘定義常數 HA=”Hello World !!”
```

爲了區隔其它的敘述、變數，因此，在習慣上我們都會將常數部份以大寫英文字母表示；另外，和變數不同的是，常數的前面不需加『$』符號。來看看底下這個例子：

```
<BODY>
    <P>檔案是 <? echo __FILE__; ?> </P>
    <P>目前是第 <? echo __LINE__; ?> 行 </P>
    <P>PHP 版本是 <? echo PHP_VERSION; ?> </P>
    <P>作業系統是 <? echo PHP_OS; ?> </P>
    <?
        define("PI", 3.1415926);
        define("HA", "Hello World !!");
    ?>
    <P>常數 PI 是 <? echo PI; ?> </P>
    <P>常數 HA 是 <? echo HA; ?> </P>
</BODY>
```

第四項　變數的變數

變數的變數是 PHP 中很有趣的一個功能，舉個例子來看比較容易理解：

```
<?
    $x=100;
    $y="x";
    $z="y";
    echo "<P>" . $x;
```

```
    echo "<P>" . $$y;                    ‘$$y 之中的$y 先展開成 x，就成爲$x
    echo "<P>" . $$$z;                   ‘同上，$$$z ➔ $$y，$$y ➔ $x
?>
```

這個例子的三個輸出結果都是 100，這是因爲『$$$z』之中的『$z=y』，因此『$$$z』就展開變成『$$y』，而其中的『$y=x』，因此『$$y』就會展開變成『$x』，而『$x=100』，因此就輸出結果是 100。
變數的變數應用在實際的程式中，可以節省很多的力氣，在未來實務操作時，一定會實際應用到的。

第五項　陣列

PHP 中的陣列和變數一樣，很自由，和其它的程式語言比較，PHP 的陣列不需事前宣告、隨時可以改變大小、陣列的索引除了數字之外還可以是字串。
陣列的功能提供我們對於需要大量使用變數的狀況，有一個很好的解決方案，我們來看看底下這個簡單的例子：

```
<?
    $a[1]=123;                        ‘這就是一個以數字爲索引的陣列
    $a[2]=456;
    $a[3]=789;
    echo "<p> $a[1]";
    echo "<p> $a[2]";
    echo "<p> $a[3]";
?>
```

底下這個例子也是一個陣列，但它是以字串爲索引的陣列，一樣可以使用，而且，就某一方面而言，它更人性化一些：

```
<?
    $a["name"]="王大";                ‘以字串爲索引的陣列
    $a["web"]="http://www.kingbig.com";
    $a["email"]="kingbig@kingbig.com";
    echo "<p>" . $a["name"];          ‘置於雙引號的外面，用字串連接方式輸出
    echo "<p>" . $a["web"];
    echo "<p>" . $a["email"];
?>
```

底下這三種設定陣列的方式（實際上應該只有二種），得到的結果是完全相同的，第二種設定陣列的方式是使用 array()函數，比較方便一些，但是要注意，使用 array()函數之前若有指定陣列的索引、值的話，都會被清除，array()函數之後再指定陣列新的索引、值，則沒有關係，最好的方式是單純用一種你習慣的方式：

```
$a[1]=123;                            ‘第一種設定陣列方式
$a[2]=456;
$a[3]=789;
```

```
$a=array( 1=>123, 2=>456, 3=>789 );   ‘第二種設定陣列方式

$a=array(                             ‘和第二種設定陣列方式一樣，也可以分行
    1=>123, 2=>456, 3=>789
    );                                ‘最後加上『;』結束
```

在使用 array()函數時，要記得在每個元素之間使用『,』分隔。

第六項　二維陣列

二維陣列使用的地方很多，例如學生成績單、通訊錄、以及將來我們使用的資料庫等，都可以用二維陣列表示，底下是一個二維陣列的例子：

```
<?
    $a[1][1]=110;                     ‘設定二維陣列
    $a[1][2]=120;
    $a[1][3]=130;
    $a[2][1]=210;
    $a[2][2]=220;
    $a[2][3]=230;

    echo "<p>" . $a[1][1];            ‘輸出二維陣列
    echo "<p>" . $a[1][2];
    echo "<p>" . $a[1][3];
    echo "<p>" . $a[2][1];
    echo "<p>" . $a[2][2];
    echo "<p>" . $a[2][3];
?>
```

我們也可以這樣子定義二維陣列：

```
<?
    $a=array(
        1=>array( 1=>110, 2=>120, 3=>130 ),        ‘千萬別忘了這裡有個逗號
        2=>array( 1=>210, 2=>220, 3=>230 )
        );

    echo "<p>" . $a[1][1];
    echo "<p>" . $a[1][2];
    echo "<p>" . $a[1][3];
    echo "<p>" . $a[2][1];
    echo "<p>" . $a[2][2];
    echo "<p>" . $a[2][3];
?>
```

三維陣列、及多維陣列的設定方式，和二維陣列的設定方式大同小異，有機會的話試試看吧！

第七項　自訂函數

使用自訂函數的功能可以將一些比較常用、具有特定功能的敘述集合起來，以便隨時可以呼叫函數，例如：

```
<?
    function double($i) {              ‘自訂一個名爲 double()的函數
        return $i*2;                   ‘將傳入的值運算後，傳回
    }

    $x=3;
    $y=double($x);                     ‘變數$y 接收函數傳回的值
    echo "<p> $x";
    echo "<p> $y";
?>
```

使用函數，通常會同時傳入某些參數，就如例子中的『$y=double($x)』敘述，是將$x 的值傳送到函數，而在『function double($i)』的敘述中，是由變數$i 來接收$x 所傳送過來的值，並且使用變數$i 在函數中運算，再透過『return』敘述將結果傳送回來。
如果說要傳遞的參數是陣列的話，其實也只需要當做一般變數使用即可，參考底下的例子：

```
<?
    function doublearray($b) {         ‘接收的陣列置於$b 之中
        $b[2][1]=$b[2][1]*2;
        $b[2][2]=$b[2][2]*2;
        $b[2][3]=$b[2][3]*2;
        return $b;                     ‘也可以將$b 陣列直接傳回
    }
    $a=array(                          ‘定義一個$a 的二維陣列
        1=>array( 1=>110, 2=>120, 3=>130 ),
        2=>array( 1=>210, 2=>220, 3=>230 )
        );
    $c=doublearray($a);                ‘傳遞$a 陣列，並由$c 接收傳回值
    echo "<p>" . $c[2][1] . ", " . $c[2][2] . ", " . $c[2][3] ;
?>
```

自訂函數時，也可以同時就設定參數的預設值，例如：

```
<?
    function sum($a, $b=60, $c=60) {  ‘設定參數的預設值
        $sum=$a+$b+$c;
        echo "總計: " . $sum;
    }
    $x=sum(80);                        只傳送一個參數
?>
```

使用函數的預設參數時，要記得將具預設參數的部份靠右，沒有預設參數的部份靠左。這是因爲我們在呼叫函數，傳入的參數值是由左邊一個一個傳送到函數之中，因此，若我們省略某些參數時，也只有函數的左邊的參數會取得傳入值，右邊的參數就會發生狀況，因此，才必須將具預設參數的部份置於右邊。

第八項　Global 宣告

在 PHP 網頁中，變數的有效範圍是在同一個 PHP 網頁的全部範圍，有需要時隨時可以拿來使用。
唯一需要注意的是函數（自訂函數）中，由於在函數之內是屬於獨立的區域，因此，在函數內的變數都是區域變數，即使在函數之中擁有和 PHP 網頁相同的變數名稱，也不會互相干擾。
反過來說，如果希望函數內與函數外的變數是同一個，就必須使用 global 宣告，來看看 global 宣告的例子：

```
<?
    function globaltest() {
        global $x;                    ‘宣告 $x 是全域變數
        echo "<P>變數 x=" . $x;       ‘$x=100
        echo "<P>變數 y=" . $y;       ‘$y 沒有指定，是空值
    }
    $x=100;
    $y=99;
    $z=globaltest();
?>
```

使用 global 宣告變數時，最好將它放在函數一開始的地方，在經過 global 宣告後，就可以隨時指定新的值給變數了。

第九項　GLOBALS 變數

若在每個函數中都要使用 global 宣告變數，那會相當麻煩，因此，PHP 提供了一個$GLOBALS（請注意大小寫）的陣列變數，讓我們可以使用比較輕鬆的方式存取 PHP 中的變數，看看底下的例子：

```
<?
    function globaltest() {
        echo "<P>變數 x=" . $GLOBALS["x"];       ‘使用變數名稱做爲陣列索引
        echo "<P>變數 y=" . $GLOBALS["y"];
    }
    $x=100;
    $y=99;
    $z=globaltest();
?>
```

使用$GLOBALS 陣列時，是以 PHP 網頁中的變數名稱做爲陣列索引，而得到的值就是原本變數的值。
若是二維陣列，就沒有辦法使用$GLOBALS 陣列，還是必須使用 global 宣告哦！

第十項　跳脫字元

變數的前面都會加『$』符號，那，如果我們希望在使用 echo 輸出資料，而其中包含『$』符號時，PHP 也會將緊接在後面的字串當成是變數。來看看底下的錯誤的狀況：

```
echo "<P>加上$的符號表示變數</P>";          '『$』符號之後的文字會被當成變數
echo "<P>這裡是"重點"，要特別注意！</P>";   '字串需包含在同一組『"』之間
```

我們必須加上跳脫字元『\』來避免這種情況發生，因此，應該修改如下：

```
echo "<P>加上\$的符號表示變數</P>";
echo "<P>這裡是\"重點\"，要特別注意！</P>";
```

有一些具有特殊意義的符號，像是『$』表示變數、『"』表字串符號，如果加上跳脫字元，可以使這些符號暫時『跳脫』原本代表的意義、功能。底下表格列出一些常用到的特殊符號：

符號	說明

\n	linefeed，換行；就像上個例子，直接使用 echo 將標籤及文字都置於其中，當我們使用瀏覽器的檢視原始檔案時，會看到那二行是列在同一行的，對於我們在除錯時，是很不方便的，因此，可以撰寫成：
	`echo "<P>加上\$的符號表示變數</P>\n";`
	`echo "<P>這裡是\"重點\"，要特別注意！</P>";`
	那麼，透過瀏覽器檢視原始檔案時，就會變成獨立的二行了。

\r	return，歸位字元；較少使用。
\t	tab，定位；使用的時機和\n 一樣。
\\	backslash，印出倒斜線。
\$	dollar sign，印出$符號。
\"	double-quote，印出雙引號。

第二節　你必須知道的功能

第一項　include

include 函數可以讓我們將某個檔案包含到目前的網頁之中，這個功能對於一般撰寫網頁有很大的幫助。例如，我們有一個檔案，它的完整內容就只有以下數行（不用包含<HEML>、<BODY>…等其它的標籤）：

```
‘檔案名稱：menu.php
<P><A href=http://www.kingbig.com>山賊老巢</A></P>
<P><A href=http://www.kingbig.com/~kingbig>山賊圖書館</A></P>
<P><A href=http://www.ez-go.net>台東易購網</A></P>
```

接下來製作首頁，其中包含二欄一列的表格，所有的資料都放置在表格之中，左邊的儲存格使用 include 包含 menu.php 這個網頁，而右邊的儲存格中置入應該要顯示的網頁內容，參考如下：

```
‘檔案名稱：index.php
<HTML>
<BODY>
    <TABLE align=center border=1 width=80%>
        <TR valign=top>                          ‘使表格的內容都靠上對齊
            <TD width=120>
                <? include(“menu.php“); ?>       ‘包含 menu.php 這個網頁
            </TD>
            <TD>                                 ‘這個儲存格輸入要顯示的內容
                <P>這裡請輸入你的內容</P>
            </TD>
        </TR>
    </TABLE>
</BODY>
</HTML>
```

這樣做完之後，可能你還沒發覺這有什麼好處，但是，如果接下來的其它幾個網頁（例如 01.php、02.php、03.php…），都依這個結構製作，左邊是 menu.php，而右邊是網頁顯示的內容，那麼，你會清楚的知道，你只需要更改 menu.php 的內容時，所有網頁左邊顯示的超連結，都會一起變更，用來建置網站，算是幫我們省了很多維護與管理的動作，而且，還可以給訪客一致性的瀏覽操作環境。

包含其它網頁的函數其實可以分成四種函數：

include( )
require( )
include_once( )

require_once( )

其中，include 與 require 最大的差異在於前者可以置於迴圈及判斷式，而後者不行，因此，include 的方式比較靈活。
至於加上『_once』的函數是 PHP4 新增功能，可以取代原本的函數，新函數在呼叫時會先判斷是否已經呼叫過了，若已經呼叫過的話，就不會再呼叫一次，建議大家使用新函數來包含檔案。

第二項　網頁重新導向

使用以下方式，可以將某一個網頁的連結，重新導向到另一個網頁：

```
‘檔案名稱：redir.php
<?
    header(“location:phpinfo.php“);
?>
```

使用 header 函數有個注意事項，不能在 header 函數之前有任何的資料輸出到瀏覽器的網頁。也就是說，不能使用 echo 輸出、不能有任何 HTML 的標籤、連空行或空格都不行，底下是個錯誤例子，是我曾犯的錯誤，花了我好久的時間才知道問題那麼簡單，希望你不要犯下同樣的錯誤：

```
                                    ←我多空了這一行，這也是輸出，不行！
<?
    header(“location:phpinfo.php“);
?>
```

第三項　網頁重新讀取

我們在之前的 HTML 部份曾經介紹可以定時重新讀取網頁的功能，PHP 也提供這個功能，一樣很簡單，參考如下：

```
<?
    header("refresh:5");
    echo date("h:i:s");
?>
```

由於也是使用 header 函數，一樣也不能在 header 之前有任何輸出動作。另外，如果希望能在重新讀取時，導向至另一個網頁，可以另外再加上 url 的敘述，參考以下例子：

```
<?
    header("refresh:5;url=http://www.kingbig.com");
    echo "五秒後連結到 山賊老巢";
```

```
?>
```

第四項　cookie

如果你希望你的網站能夠和訪客之間有良好的互動，那你就一定得知道『cookie』。簡單的說，cookie 就是讓 server 端的電腦在你的電腦中存入一些資料、也可以取出這些資料，這些資料是很小量，而且，也只能存取自己所建立的 cookie 部份。
在 PHP 中建立 cookie、以及取出 cookie 的資料，都幫我們簡化很多，我們來看個例子，就很容易明白 cookie 的操作。
form2.htm 網頁在於建立登錄的表單，並將登入的資料傳送到 cookie1.php 網頁：

```
‘檔案名稱：form2.htm
<HTML>
<BODY>
    <H1>登入表單</H1>
    <FORM method=post action=cookie1.php>
        <p>帳號： <INPUT type=text name=id> </p>
        <p>密碼： <INPUT type=password name=passwd> </p>
        <p><INPUT type=submit value="登入">    <INPUT type=reset value="清除">
    </FORM>
</BODY>
</HTML>
```

cookie1.php 網頁原本的功能應該是要先判斷輸入的帳號（變數$id）及密碼（變數$passwd）正確時，才將資料透過 setcookie 函數存成訪客的 cookie，但由於我們還沒有學到條件的判斷及資料庫存取，就暫時略過條件判斷的部份，直接將傳過來的$userid 變數，存成名稱是 loginname 的 cookie：

```
‘檔案名稱：cookie1.php
<?
    setcookie("loginname",$id);        ‘存成名爲 loginname 的 cookie
    header("location:cookie2.php");    ‘任務完成，將網頁導向至 cookie2.php
?>
```

cookie2.php 網頁的功能，是實際的將存入的 cookie，提出來使用。我們所存的 cookie 名稱是 loginname，所以要提出來使用時，就使用$loginname 的變數，就這麼簡單：

```
‘檔案名稱：cookie2.php
<?
    echo "哈囉, $loginname 你好！歡迎你的光臨！";
?>
```

第五項　設定有效期限的 cookie

前面練習的 cookie 有效期限，是在瀏覽器開啓的時候，如果瀏覽器關閉結束，那麼，當時所建立的 cookie 也會同時結束。如果有需要，我們可以延長 cookie 的有效期限，例如底下的例子，可以記錄訪客瀏覽這個網頁的次數：

```
<?
    setcookie("times",$times+1,time()+86400);  '提出 cookie 後加 1，再存回去
?>
<HTML>
<BODY>
    <p>這是你第 <? echo $times; ?> 次光臨！</p>
</BODY>
</HTML>
```

cookie 的有效期限是以秒爲單位，因此，上面的例子有效期限是 24 小時內，只要訪客在 24 小時內來訪，就可以顯示訪客來訪的次數，同時再將有效期限延長 24 小時。
如果訪客是在 24 小時後才再次瀏覽這個網頁，之前的記錄會失效，再重新來過，因此，你可以視情況決定，要將 cookie 的有效期限設定多久。

第六項　cookie 的限制

一個 cookie 儲存資料的最大容量是 4KB，一台伺服器最多可以在訪客的電腦中建立 20 個 cookie，所有伺服器所建立的 cookie 總數最多是 300 個。
訪客瀏覽網頁時所建立的 cookie 是存放在記憶體中，當訪客關閉瀏覽器時，cookie 也同時結束，至於尚未到期的 cookie 就會被儲存在 cookies.txt 的檔案中（使用 win2000 的話則是放在使用者目錄的 cookies 目錄中）。
我們去檢視這些檔案時，會發現它是以『使用者名稱@網域名稱』方式儲存，類似電子郵件的格式，不過，這當然不是我們的電子郵件，而是那些留下 cookie 的伺服器的網域名稱，換句話說，某個伺服器也只允許存取自己所建立的 cookie，不能存取其它伺服器所建立的 cookie 資料。

第七項　檔案上傳

檔案上傳，很棒吧！而且，還很簡單。設計檔案上傳的表單，有幾點要注意的：

1. 表單的 method 要用 post
2. 在<FORM>標籤中要再加上『enctype=”multipart/form-data”』的敘述
3. 使用『type=file』的輸入欄位，就會有一個『瀏覽』按鈕，選擇使用者硬碟中的資料，其中『name=upfile』中的 upfile 是我們給這個輸入欄位所建立的欄位名稱

upload.htm 網頁負責建置使用者瀏覽、選取檔案的表單：

```
'檔案名稱：upload.htm
<HTML>
<BODY>
    <H1>上傳檔案</H1>
    <FORM method=post action=upload.php enctype="multipart/form-data">
        <p>選取檔案： <INPUT type=file name=upfile> </p>
        <p><INPUT type=submit value="傳送">
    </FORM>
</BODY>
</HTML>
```

upload.php 是負責接收檔案，並將上傳的檔案儲存在『/tmp』目錄中，而且是以『$upfile』爲檔案名稱，這並不是原來的檔名，而是 PHP 暫時建立的檔名，而且，會在這個 PHP 網頁執行結束時，就將檔案移除，因此，我們必須在 PHP 網頁執行結束前，得先將檔案複製備份才行。
透過$upfile_name、$upfile_size 這二個變數（依原本欄位名稱而產生的二個變數），可以取得檔案真正的檔名、及檔案大小，再使用 copy 函數就可以將檔案複製下來了。

```
'檔案名稱：upload.php
<?
    echo "<P> $upfile";              'PHP 建立的暫時檔名
    echo "<P> $upfile_name";         '檔案真正的檔案名稱
    echo "<P> $upfile_size";         '檔案的大小，以 byte 爲單位
    copy($upfile, "upload/$upfile_name");
?>
```

我使用『copy($upfile, "upload/$upfile_name");』敘述，是將$upfile 檔案複製到 upload.php 這個網頁相對的 upload 目錄中，並且使用$upfile_name 將檔案重新命名爲原來的檔案名稱。
你也可以使用絕對路徑來複製檔案，例如『copy($upfile, "/tmp/$upfile_name");』。無論是使用相對路徑或絕對路徑來複製檔案，該目錄的權限都要設成 777，檔案才能存進去。

檔案上傳的限制

在『/etc/php.ini』之中有限制上傳檔案的大小是 2MB，你可以自行調整。開啓『/etc/php.ini』檔案之後，可以找到以下二行的內容：

```
; Maximum size of POST data that PHP will accept.
post_max_size = 8M

; Maximum allowed size for uploaded files.
upload_max_filesize = 2M
```

依你的需求調整 upload_max_filesize 的值就可以了。

第三節　運算符號

底下列出一些常用到的運算符號，和其它的程式語言比較，絕大多數都是類似的。

第一項　算術運算

符號	說明
+	加號；例如：$a + $b
-	減號；例如：$a - $b
*	乘號；例如：$a * $b
/	除號；例如：$a / $b
%	餘數；例如：$a % $b

第二項　位元運算

針對變數的『位元』做運算，請參考底下二個表格：

符號及範例	說明
$a & $b	and；當二個位元皆爲 1 時，結果爲 1，否則爲 0
$a | $b	or；二個位元只要有一個是 1，結果爲 1，二個都是 0 時，結果才是 0
$a ^ $b	xor；二個位元不同，結果是 1，如果二個都是 0、或都是 1，則結果爲 0
~ $a	not；反相，原來是 1 則變爲 0，原來是 0 則變爲 1

由於這些運算是針對變數的位元來處理，因此，我們需要將變數放大，來看看它的 0 與 1 之間運算的結果，假設 A、B 分別代表 0 或 1 的狀況，我們可以得到如下的表格（通常稱爲真值表）：

A	B	Not A	A And B	A Or B	A Xor B
1	1	0	1	1	0
1	0	0	0	1	1
0	1	1	0	1	1
0	0	1	0	0	0

第三項　比較運算

主要是用在條件式的判斷、或迴圈流程的判斷，以控制程式的運作。這些比較式所得到的結果都是『成立』、『不成立』二種結果。

符號	說明
$a == $b	等於，是二個等號；比較$a 是否等於$b 它和『=』不同，一個等號代表的是指定，也就是將右方的值指定給左方的變數，例如『$a=$b』是將變數$b 的值指定給變數$a，因此，$a、$b 二個變數的值會是一樣的。至於二個等號是表示『是否相等』，因此得到的結果會是『相等（成立）』、或『不相等（不成立）』。

$a === $b	等於，三個等號，只適用於 PHP4；比較$a 是否等於$b
	除了會比較『值』是否相等之外，還會比較『型別』是否相等。例如：
	`$a=123;`
	`$b="123";`
	那麼，『$a==$b』的結果是成立，而『$a===$b』的結果是不成立，前者是因爲 PHP 自動轉換型別，所以比較的『值』是一樣的，而後者因爲『型別』不同，所以結果就是不成立。

$a != $b	不等於；比較$a 不等於$b
$a !== $b	不等於，適用於 PHP4；比較$a 與$b 的值不等於，而且連型別也不同
$a < $b	小於；比較$a 小於$b
$a > $b	大於；比較$a 大於$b
$a <= $b	小於等於；比較$a 小於或等於$b
$a >= $b	大於等於；比較$a 大於或等於$b

第四項　邏輯運算

邏輯運算主要在判斷多個運算式比較的結果爲何。實際上，前面提到的『位元運算』也就是邏輯運算，是針對位元，而底下表格的邏輯運算是對於前後的結果來比較，只需要將前面提到的真值表之間的 1 與 0 分別代換成『True』與『False』。

A	B	! A	A And B A && B	A Or B A || B	A Xor B
T	T	F	T	T	F
T	F	F	F	T	T
F	T	T	F	T	T
F	F	T	F	F	F

舉個例子，假設：

```
有三個變數：$a=10, $b=12, $c=14
而運算式是：($a<$b) and ($b<$c)➔（成立） and （成立）➔結果是『成立』
```

第四節　流程控制

流程控制和大部份程式語言的方式都差不多，看個幾次，相信很容易就會熟悉的。

第一項　if

條件式的判斷，可以說是最常用到的功能，換句話說，就是『你一定得學起來』。來看看最簡單的敘述，舉個例子：

```
if ($a > $b) echo "a 大於 b";
    或者，也可以寫成這樣
if ($a > $b)
    echo "a 大於 b";
```

這個敘述的意思，你可以解釋成『假如變數 a 大於變數 b 的話，就列印出“a 大於 b”』，那，如果 a 沒有大於變數 b 的話，當然就不會列印了。
這就是條件式在比較時，會有『成立』、或『不成立』的二種不同狀況，以及是否執行後面的敘述。
會了嗎？就是這麼簡單。

再來看看另一個例子：

```
if ($a > $b) {
    echo " <p> a 大於 b";
    echo " <p> a=$a, b=$b";
}
```

這個例子在條件式成立時，做了比較多的事情，因此使用『{…}』將這些敘述包含著。你可以解釋成『假如變數 a 大於變數 b 的話，就列印出“a 大於 b”，而且繼續列印“a=$a, b=$b”』同樣的，如果變數 a 沒有大於變數 b 的話，這些敘述就都不會被執行囉！

第二項　if...else

再來看個比較完整的 if 使用方式，如下：

```
if ($a > $b) {
    echo "a 大於 b";
```

```
} else {                                    ‘加上了 else 的敘述
    echo "a 沒有大於 b";
}
```

我們可以解釋成『假如變數 a 大於變數 b 的話，就列印出“a 大於 b”，如果沒有的話，就列印“a 沒有大於 b”』。
加上 else 敘述之後，我們就同時可以處理條件式『成立』、與『不成立』時的狀況，是很完整的 if 使用方式了。

第三項　if...elseif...else

如果還有更多的條件要判斷時，還可以這樣子：

```
if ($a > $b) {                              ‘條件式 1
    echo "a 大於 b";
} elseif ($a == $b) {                       ‘條件式 2，使用 elseif 敘述
    echo "a 等於 b";
} else {                                    ‘前面條件都沒有符合時
    echo "a 小於 b";
}
```

我們可以解釋成『假如變數 a 大於變數 b 的話，就列印出“a 大於 b”，又，如果變數 a 等於變數 b 的話，就列印出“a 等於 b”，都沒有條件符合的話，就列印“a 小於 b”』。
elseif 可以增加不同條件式的判斷，只要程式有需要增加條件的判斷，我們可以再繼續增加 elseif 的判斷。

第四項　while

只要條件式成立，就執行迴圈。來看看底下的例子：

```
$i = 1;
while ($i <= 10) {
    echo $i;
    $i=$i+1;
}
```

第五項　do...while

和前述的 while 相同，只要條件式成立，就執行迴圈，但是，由於條件的判斷是放在後面，因此，就算條件式不成立，其中所包含的敘述還是會先執行一次。

```
$i = 0;
```

```
do {
    echo $i;                          ‘先列印 $i
} while ($i>0);                       ‘再來判斷條件式，是否要繼續執行
```

第六項　for

在 for 迴圈中，我們必須給它三個條件：初始值、條件判斷、步進值。其實和 while 敘述很類似，來看個例子：

```
for ($i = 1; $i <= 10; $i=$i+1) {     ‘分別設定 初始值、條件判斷、步進值
    echo $i;
}
```

- 『$i = 1』是初始值，設定變數$i 從 1 開始。
- 『$i <= 10』是條件判斷，當條件式成立時，就繼續執行迴圈。
- 『$i = $i + 1』是步進值，每執行迴圈一次，就將變數$i 加 1。
- 三個敘述之間要使用分號隔開。

很多程式在介紹到 for 迴圈時，都會以九九乘法表做個例子，我們也來試試看吧！由於在網頁上不容易定位，因此我們就用表格來做列印定位的動作：

```
<?
    echo "<table width=90% align=center border=1>";
    echo "<tr>";

    for ($i = 1; $i <= 9; $i=$i+1) {
        echo "<td>";                          ‘總共建立九個儲存格
        for ($j = 1; $j <= 9; $j=$j+1) {      ‘每一個儲存格，都存放九行內容
            echo "$i * $j = " . $i*$j . "<br>" ;
        }
        echo "</td>";
    }

    echo "</tr>";
    echo "</table>";
?>
```

第七項　foreach

這是 PHP4 新增的功能，讓我們可以很輕鬆使用陣列的值、或索引運用在迴圈之中。foreach 迴圈的原型有以下二種方式：

```
foreach(array_expression as $value) statement          ‘只取出值
foreach(array_expression as $key => $value) statement  ‘取出值與索引
```

其中的 array_expression 表示某個陣列，$value 表示陣列其中的值，$key 表示陣列索引。foreach 迴圈是判斷陣列中有多少個索引（或者是有多少個值），就會由第一個開始，執行多少次迴圈。來看個例子：

```
<?
    $b=array( 1=>110, 2=>120, 3=>130 );    ‘建立陣列

    foreach($b as $bkey => $bvalue) {      ‘由陣列中取出 key 及 value
        echo "<P>\$b 陣列, 索引=$bkey, 值=$bvalue";
    }
?>
```

$b 陣列由於有三個索引（也有三個值），所以 foreach 會執行三次，分別取出陣列中的索引及元素的值。

第八項　switch

switch 和 if 敘述非常類似，可以依條件不同而執行不同的動作，不同的是，if 敘述在比較、判斷時都是只有『成立』與『不成立』二種選擇，雖然可以使用 elseif 敘述增加判斷的條件，讓程式的流程有更多的選擇，但是，在同一個時間，程式的流向仍然只有二種選擇。
使用 switch 的話，可以讓程式在選擇時，有更多的『路』可以走。舉個例子，底下程式可以在一個星期之中，每天讓訪客看到的網頁都是不同的：

```
<?
    $i=date("w");                    ‘由日期中取得今天是星期幾？
    switch ($i) {
        case 0:                      ‘星期日
            header("location:week00.php");
            break;
        case 1:                      ‘星期一
            header("location:week01.php");
            break;
        case 2:                      ‘星期二
            header("location:week02.php");
            break;
        case 3:                      ‘星期三
            header("location:week03.php");
            break;
        case 4:                      ‘星期四
            header("location:week04.php");
            break;
        case 5:                      ‘星期五
            header("location:week05.php");
            break;
        default:                     ‘上面都不符合時，一定是星期六
```

```
            header("location:week06.php");
    }
?>
```

如果你懶得做那麼多不同的網頁，那還可以每二天換一個網頁：

```
<?
    $i=date("w");                    ‘由日期中取得今天是星期幾？
    switch ($i) {
        case 0:                      ‘星期日、星期一
        case 1:
            header("location:week00.php");
            break;
        case 2:                      ‘星期二、星期三
        case 3:
            header("location:week02.php");
            break;
        case 4:                      ‘星期四、星期五
        case 5:
            header("location:week04.php");
            break;
        default:                     ‘上面都不符合時
            header("location:week06.php");
    }
?>
```

使用 switch 時，要注意 break 敘述，如果沒有加上的話，程式流程是會繼續向下一個 case 執行的，例如底下的例子，就會由 case 1 開始執行到最後一個：

```
<?
    $i=1;
    switch ($i) {
        case 0:
            echo "0";
        case 1:
            echo "1";
        case 2:
            echo "2";
        default:
            echo "other";
    }
?>
```

第四章　學習 MySQL 也很容易

第一節　MySQL 認識

MySQL 是一個快速、多執行緒、多使用者和穩定的 SQL 資料庫伺服器。在 Unix 和 OS/2 上使用 MySQL 是免費的，而 Windows 系統上必須取得授權。
MySQL 的官方發音是『My Ess Que Ell』，大家都習慣唸做 MY-SEQUEL。

在之後章節常會使用 db_name(資料庫名稱)、tbl_name(資料庫名稱)和 col_name（欄位名稱）。例如：

```
mysql> SELECT col_name FROM db_name.tbl_name;
```

在 MySQL 中使用大小寫是相同的，並沒有區別。

第一項　使用 MySQL 的優點

1. 使用核心執行緒的真正多執行緒，也就是說，如果你的系統是多 CPU 它也能輕易使用。
2. 可使用 C, C++, Java, Perl, PHP, Python…等不同的程式。
3. 可在不同的平台上使用。
4. 多種欄位的資料格式。
5. 使用最佳化的 one-sweep multi-join 可以非常快速的做聯結（join）。
6. 在 SELECT 及 WHERE 敘述部份支援全部的運算子及函數。
7. 經過高度最佳化的 SQL 函數可以很快速的取得我們要的資料，而且，在查詢初始化之後，應該不會再配給記憶體來使用了。
8. 支援完整的 GROUP BY、ORDER BY 子句。
9. 在一個查詢之中可以使用不同的資料庫。
10. 控制大型的資料庫。例如我們在使用包含了 50,000,000 筆記錄的資料庫。

其它還有非常多的優點…

第二項　MySQL 的命名

MySQL 的命名是使用由 3 個數字和一個字尾組成的版本號碼。例如，一個像 mysql-3.21.17-beta 的版本號這樣解釋：

- 第 1 數字(3)描述文件格式。所有版本 3 的發行都有相同的檔案格式。當一個版本 4 出現時，每個資料庫都將必須轉換到新格式。

- 第 2 數字(21)是發行版本。通常有 2 種選擇，一個是發行穩定的版本(目前是 22)，而另外還有一個開發中的版本(目前是 23)。通常兩者都是穩定的，但是開發中版本可能會有些問題，例如缺乏新功能的說明或是在某些系統上編譯失敗。
- 第 3 個數字(17)是在此發行級別的修定版本。
- 最後一個字尾是顯示發行版本的穩定性級別，可能的字尾有：
  - alpha 表示發行版本中包含大量的新程式碼，而這些新的程式碼沒有經過 100%測試。
  - beta 意味著所有的新程式碼被測試了，沒有新增其它功能，而且應該也沒有已知的錯誤存在。
  - gamma 是一個發行了一段時間的 beta 版本，應該是可以正常運作。這也是其他公司稱爲一個 release 的版本。
  - 如果沒有加上其它的字尾，這就是一個穩定的版本。

第三項　以 RPM 檔案安裝 MySQL

以 RPM 檔案來安裝 MySQL，我們可能會需要的檔案有：

檔案	說明
MySQL-VERSION.i386.rpm	MySQL server，必須安裝。
MySQL-client-VERSION.i386.rpm	標準的 MySQL client 端程式，必須安裝。
MySQL-bench-VERSION.i386.rpm	測試及效能測試的程式，選擇性安裝。
MySQL-devel-VERSION.i386.rpm	函式庫，以及包含你所需要編譯的其它 MySQL clients 端程式，例如 Perl 模式。
MySQL-VERSION.src.rpm	包含以上所有 RPM 檔案的原始程式。

若只需要最小化安裝的方式，可以執行：

```
shell> rpm -ivh MySQL-VERSION.i386.rpm
shell> rpm -ivh MySQL-client-VERSION.i386.rpm
```

MySQL 所有的資料都會儲存於/var/lib/mysql 的目錄中。

第二節　MySQL 資料庫基本操作

安裝 MySQL 之後，記得要先執行 mysqladmin 指令，建立密碼，例如：

```
mysqladmin -u root password '1234'
```

建立的密碼請牢牢記住，將來的資料庫建立、查詢、PHP 連結…都會需要使用這個密碼。

第一項　mysqlshow

mysqlshow 指令可以瀏覽資料庫的結構，例如底下的指令是檢視目前的 database：

```
[root@vmlinux /root]# mysqlshow -p    ‘顯示目前的資料庫
Enter password:****
+-----------+
| Databases |
+-----------+
| mysql     |
| test      |
+-----------+
```

而底下的指令可以檢視 mysql 資料庫的 table：

```
[root@vmlinux /root]# mysqlshow -p mysql   ‘顯示 mysql 資料庫的表單
Enter password:****
Database: mysql
+--------------+
|    Tables    |
+--------------+
| columns_priv |
| db           |
| func         |
| host         |
| tables_priv  |
| user         |
+--------------+
```

第二項　資料庫

資料庫中最小的單位是『欄位（field）』，由一個一個的欄位組成一列的『記錄（record）』，再由一列一列的記錄組成『資料表（table）』，很多的資料表就組成『資料庫（database）』。
底下表格內容就是一個資料表，每一列就是一筆記錄，而每一筆記錄是由五個欄位所組成的。

no	name	class1	class2	class3
1	陳一	45	45	50
2	孫二	20	25	25

3	張三	60	85	68
4	李四	35	75	45
5	王五	85	90	86

第三項　連結 MySQL

使用『mysql -p』指令可以連結到 MySQL 資料庫，例如：

```
[root@vmlinux /root]# mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11 to server version: 3.23.32

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql>
```

要離開 MySQL，可以使用 exit、quit 指令、或者是按下『Ctrl + d』，就可以中斷 MySQL 資料庫的連結：

```
mysql> quit
Bye
```

第四項　MySQL 的一般操作

輸入指令

在 MySQL 中操作，提示字示是『mysql>』，我們可以在提示字元後面再加上指令，例如『select user();』指令，會顯示使用 MySQL 的使用者是誰：

```
mysql> select                          'MySQL 的指令可以分行
    -> user()
    -> ;                               '最後要以『;』結束
+----------------+                     '這裡開始是顯示的內容
| user()         |                     '欄位資料
+----------------+
| root@localhost |                     '欄位內容
+----------------+
1 row in set (0.01 sec)
```

取消指令

如果要取消所輸入的指令，可以使用『\c』來中斷輸入的指令。

提示符號

至於底下表格列出提示符號可能出現的型態：

提示字元	說明
mysql>	等待新的指令
->	多行指令時的提示符號
'>	表示前一個指令還缺少右方的單引號
">	表示前一個指令還缺少右方的雙引號

範例

底下列出幾個範例：

```
mysql> select user(),current_date();
+----------------+----------------+
| user()         | current_date() |
+----------------+----------------+
| root@localhost | 2001-08-26     |
+----------------+----------------+
1 row in set (0.00 sec)
```

```
mysql> select user();select current_date();
+----------------+
| user()         |
+----------------+
| root@localhost |
+----------------+
1 row in set (0.00 sec)

+----------------+
| current_date() |
+----------------+
| 2001-08-26     |
+----------------+
1 row in set (0.00 sec)
```

第五項　檢視狀態

在 MySQL 提示符號下，可以使用 show 的關鍵字來檢視資料庫、資料表、欄位

的狀態及資訊，這個功能和在 Linux shell 直接使用 mysqlshow 的方式很類似。

檢視資料庫

```
mysql> SHOW DATABASES;
+----------+
| Database |
+----------+
| mysql    |
| mytest   |
| test     |
+----------+
4 rows in set (0.00 sec)

mysql> SHOW DATABASES LIKE 't%';
+---------------+
| Database (t%) |
+---------------+
| test          |
+---------------+
1 row in set (0.00 sec)
```

檢視資料表

```
mysql> SHOW TABLES;                    ‘檢視預設資料庫的資料表
+-----------------+
| Tables_in_mytest |
+-----------------+
| t1               |
| t2               |
+-----------------+
2 rows in set (0.01 sec)

mysql> SHOW TABLES FROM mytest;        ‘加上 from 檢視某個資料庫的資料表
+-----------------+
| Tables_in_mytest |
+-----------------+
| t1               |
| t2               |
+-----------------+
2 rows in set (0.01 sec)
```

檢視欄位

```
mysql> SHOW COLUMNS FROM t2;
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| no         | int(11)  | YES  |     | NULL    |       |
| name       | char(8)  | YES  |     | NULL    |       |
| phone      | char(16) | YES  |     | NULL    |       |
| address    | char(60) | YES  |     | NULL    |       |
| webaddress | char(32) | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
5 rows in set (0.01 sec)

mysql> SHOW COLUMNS FROM t2 FROM mytest;  '也可以加上某資料庫的某資料表
+------------+----------+------+-----+---------+-------+
| Field      | Type     | Null | Key | Default | Extra |
+------------+----------+------+-----+---------+-------+
| no         | int(11)  | YES  |     | NULL    |       |
| name       | char(8)  | YES  |     | NULL    |       |
| phone      | char(16) | YES  |     | NULL    |       |
| address    | char(60) | YES  |     | NULL    |       |
| webaddress | char(32) | YES  |     | NULL    |       |
+------------+----------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

第五章　動手建立資料庫

第一節　資料庫、資料表、欄位的命名

資料庫、資料表、欄位的命名規則，和一般檔案、目錄的命名類似，可以使用英文、數字，特殊符號不能使用，另外也不能使用『\』、『.』這些符號，最大可以命名到 64byte。
在 MySQL 中是沒有分英文的大小寫的，但是，資料庫及資料表在命名、使用時，是有分英文的大小寫，這個原因是 Linux 的檔案系統是有分大小寫的。
假設我們建立了一個『mytest』的資料庫，會在 Linux 系統中產生一個『/var/lib/mysql/mytest』的目錄，若在『mytest』資料庫中建立一個『t1』的資料表，相對的會在『/var/lib/mysql/mytest』目錄中產生一些以『t1』開頭的檔案。
查詢時，可以使用『col_name』、『tbl_name.col_name』、『db_name.tbl_name.col_name』，要使用那一種方式，可以視狀況來決定。

第二節　資料庫

MySQL 原本包含了二個資料庫：mysql、test。
Test 資料庫是讓我們測試用的，而 mysql 資料庫是記錄使用者權限（例如是否允許建立資料庫或刪除資料庫），這個資料庫很重要不能刪除。
至於我們自己要使用的資料庫，當然就由我們自己建立了。

第一項　建立資料庫

使用『CREATE DATABASE db_name』可以建立新的資料庫：

```
mysql> create database mytest;          '建立名爲 mytest 的資料庫
Query OK, 1 row affected (0.01 sec)
```

要特別注意 db_name 的大小寫，雖然 MySQL 的指令是沒有區分大小寫，但是 db_name 及 tbl_name 是有區分大小寫的，例如：

```
mysql> create database MyTest;          '另外建立了一個 MyTest 的資料庫
Query OK, 1 row affected (0.01 sec)
```

第二項　刪除資料庫

上面的例子，我多建立了一個 MyTest 的資料庫，我若要刪除的話，使用『DROP

DATABASE db_name』指令：

```
mysql> drop database MyTest;          ‘刪除 MyTest 的資料庫
Query OK, 0 rows affected (0.00 sec)
```

第三項　使用資料庫

由於在 MySQL 中會有很多個資料庫同時存在，因此我們在使用時（例如建立資料表、新增記錄…）要先指定使用的資料庫：

```
mysql> use mytest;                    ‘指定使用的資料庫是 mytest
Database changed
```

第三節　資料表

第一項　建立資料表

使用『CREATE TABLE tbl_name (col_name…)』可以建立資料表，在 (…) 中的是資料表的欄位名稱及欄位型別，我們實際建立一個學生成績的資料表試試看：

```
mysql> create table t1 (              ‘建立 t1 資料表
    -> no integer,                    ‘學號，integer 型別
    -> name char(8),                  ‘姓名，char 型號，8 個字元
    -> class1 integer,                ‘科目 1，integer 型別
    -> class2 integer,                ‘科目 2，integer 型別
    -> class3 integer                 ‘科目 3，integer 型別
    -> );                             ‘以分號結束
Query OK, 0 rows affected (0.01 sec)
```

第二項　刪除資料表

使用『DROP TABLE tbl_name』可以刪除資料表。

第四節　記錄的新增、刪除、更新、查詢

第一項　新增記錄

使用『INSERT INTO tbl_name (col_name…) VALUES (欄位的資料)』，可以新增記錄，在 VALUES 後面所新增的值，會一個對一個的對應所代表的 col_name。如果新增的資料是所有欄位都有、而且也都依原本資料表的欄位順序時，就可以省略(col_name…)的部份。
底下的例子，第一個敘述和其它敘述的結果一樣：

```
mysql> insert into t1 (no,name,class1,class2,class3) values (1,'陳一',45,45,50);
Query OK, 1 row affected (0.01 sec)

mysql> insert into t1 values (2,'孫二',20,20,25);
Query OK, 1 row affected (0.01 sec)

mysql> insert into t1 values (3,'張三',60,85,68);
Query OK, 1 row affected (0.01 sec)

mysql> insert into t1 values (4,'李四',35,75,45);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (5,'王五',85,90,86);
Query OK, 1 row affected (0.01 sec)

mysql> insert into t1 values (6,'錢六',35,65,45);
Query OK, 1 row affected (0.01 sec)
```

第二項　查詢記錄

使用『SELECT』指令可以查詢資料，最簡單的方式是使用『SELECT * FROM tbl_name』，例如：

```
mysql> select * from t1;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    1 | 陳一 |     45 |     45 |     50 |
|    2 | 孫二 |     20 |     20 |     25 |
|    3 | 張三 |     60 |     85 |     68 |
|    4 | 李四 |     35 |     75 |     45 |
|    5 | 王五 |     85 |     90 |     86 |
|    6 | 錢六 |     35 |     65 |     45 |
```

```
+------+------+--------+--------+--------+
6 rows in set (0.00 sec)
```

使用『select * from t1;』會顯示 t1 資料表中的所有記錄。

加上 where 敘述

在使用 select 查詢資料時，可以加上『WHERE』敘述來篩選某些資料，例如我們只想要看學號 6 的成績資料時，參考如下指令：

```
mysql> select * from t1 where no=6;   '篩選出 no 欄位等於 6 的記錄
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    6 | 錢六 |     45 |     65 |     45 |
+------+------+--------+--------+--------+
1 row in set (0.01 sec)
```

第三項　更新記錄

更新記錄時，使用『UPDATE tbl_name SET col_name=新的值』敘述將資料更新，但是由於沒有指定更新的記錄是那一個，所以 MySQL 會將該 tbl_name 中所有記錄的 col_name 欄位資料都同時更新資料，千萬要小心使用。
實際操作看看：

```
mysql> update t1 set class1=class1+10;     '將 class1 欄位全部加 10
Query OK, 6 rows affected (0.00 sec)
Rows matched: 6  Changed: 6  Warnings: 0

mysql> select * from t1;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    1 | 陳一 |     55 |     45 |     50 |
|    2 | 孫二 |     30 |     20 |     25 |
|    3 | 張三 |     70 |     85 |     68 |
|    4 | 李四 |     45 |     75 |     45 |
|    5 | 王五 |     95 |     90 |     86 |
|    6 | 錢六 |     65 |     65 |     45 |
+------+------+--------+--------+--------+
6 rows in set (0.00 sec)
```

所以，更新資料時，通常會加上 where 來篩選條件，例如我們只要更新座號 6 的 class1 分數時，就可以這樣做：

```
mysql> update t1 set class1=35 where no=6;
```

```
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from t1 where no=6;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    6 | 錢六 |     35 |     65 |     45 |
+------+------+--------+--------+--------+
1 row in set (0.00 sec)
```

第四項　刪除記錄

使用『DELETE FROM tbl_name WHERE 條件式』敘述可以刪除記錄，例如：

```
mysql> delete from t1 where no=6;
Query OK, 1 row affected (0.00 sec)
```

第五項　Select 的應用

使用『select * from t1;』的敘述可以將資料表中所有的記錄都列出來，如果我們只要列出其中的座號及姓名時，可以這樣做：

```
mysql> select no,name from t1;        '直接輸入欲顯示的欄位資料
+------+------+
| no   | name |
+------+------+
|    1 | 陳一 |
|    2 | 孫二 |
|    3 | 張三 |
|    4 | 李四 |
|    5 | 王五 |
+------+------+
5 rows in set (0.00 sec)
```

order by

在 select 敘述中加上 order by，可以使列出的資料排序，例如：

```
mysql> select * from t1 ORDER BY class1;  '依 class1 排序
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
```

```
|    2 | 孫二 |     30 |     20 |     25 |
|    4 | 李四 |     45 |     75 |     45 |
|    1 | 陳一 |     55 |     45 |     50 |
|    3 | 張三 |     70 |     85 |     68 |
|    5 | 王五 |     95 |     90 |     86 |
+------+------+--------+--------+--------+
5 rows in set (0.00 sec)
```

排序時，預設是使用『遞增排序』，如果要使用『遞減排序』時，可以再加上『desc』的關鍵字，例如：

```
mysql> select * from t1 order by class1 DESC;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    5 | 王五 |     95 |     90 |     86 |
|    3 | 張三 |     70 |     85 |     68 |
|    1 | 陳一 |     55 |     45 |     50 |
|    4 | 李四 |     45 |     75 |     45 |
|    2 | 孫二 |     30 |     20 |     25 |
+------+------+--------+--------+--------+
5 rows in set (0.00 sec)
```

where 加上 order by

我們學會的 where 與 order by 二個功能可以一起運用，但是 where 一定要在前面，order by 一定要在後面，例如：

```
mysql> select * from t1 WHERE class1>=60 ORDER BY class1 desc;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    5 | 王五 |     95 |     90 |     86 |
|    3 | 張三 |     70 |     85 |     68 |
+------+------+--------+--------+--------+
2 rows in set (0.01 sec)
```

where 加上 like

這個大多應用在搜尋字串資料。之前使用 where 來篩選資料時，都是某一欄位資料等於多少、或是大於等於多少，但是，有時候我們要在一堆文字中尋找某些符合的字串時，就可以將 where 配合 like 一起應用。
例如底下範例中，我找出 class1 分數中有 5 的，都列出來：

```
mysql> select * from t1 WHERE class1 LIKE '%5%';
+------+------+--------+--------+--------+
```

```
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    1 | 陳一 |     55 |     45 |     50 |
|    4 | 李四 |     45 |     75 |     45 |
|    5 | 王五 |     95 |     90 |     86 |
+------+------+--------+--------+--------+
3 rows in set (0.01 sec)
```

其中的『%』是萬用字元，可以代替 N 個字元，另外有一個萬用字元是『_』，可以代替一個字元。

Where 加上 between…and…

主要是用來篩選某一個範圍的資料，例如底下範例是顯示 class1 的分數在 40 至 80 之間：

```
mysql> select * from t1 where class1 BETWEEN 40 AND 80;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    1 | 陳一 |     55 |     45 |     50 |
|    3 | 張三 |     70 |     85 |     68 |
|    4 | 李四 |     45 |     75 |     45 |
+------+------+--------+--------+--------+
3 rows in set (0.00 sec)
```

where 加上 and, or

在 where 敘述中可以加上 and, or 來使用多個條件式，底下的範例結果和使用『between…and…』的結果會一樣：

```
mysql> select * from t1 where class1>=40 and class1<=80;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    1 | 陳一 |     55 |     45 |     50 |
|    3 | 張三 |     70 |     85 |     68 |
|    4 | 李四 |     45 |     75 |     45 |
+------+------+--------+--------+--------+
3 rows in set (0.01 sec)
```

若純粹以上述二種不同的敘述而產生相同的結果而論，使用『between…and…』的速度要快些，但是使用 and, or 的方式，可以使你的查詢更靈活，很多狀況並不是『between…and…』可以處理，往往還是需要使用 and, or 來得到我們想要的結果。

limit

加上 select 敘途的最後面，可以限定擷取資料的筆數，有二種方式，例如：

```
mysql> select * from t1 limit 3;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    1 | 陳一 |     45 |     45 |     50 |
|    2 | 孫二 |     20 |     20 |     25 |
|    3 | 張三 |     60 |     85 |     68 |
+------+------+--------+--------+--------+
3 rows in set (0.00 sec)
```

上述方式是取出資料的前三筆記錄。另外，我們也可以取出特定位置的某幾筆記錄，例如，底下的敘述可以由第 2 筆記錄開始，連續取出 3 筆記錄，另外，記錄是由第 0 筆開始起算的：

```
mysql> select * from t1 limit 2,3;
+------+------+--------+--------+--------+
| no   | name | class1 | class2 | class3 |
+------+------+--------+--------+--------+
|    3 | 張三 |     60 |     85 |     68 |
|    4 | 李四 |     35 |     75 |     45 |
|    5 | 王五 |     85 |     90 |     86 |
+------+------+--------+--------+--------+
3 rows in set (0.00 sec)
```

第六項　修改 MySQL 資料表結構

資料庫的建置初期，總會有些考慮不夠周詳的狀況，或者是資料庫擴大時會有需要對資料表做些欄位結構的調整，我們就可以使用以下方式：

ALTER TABLE tbl_name 動作

底下就對可能遇到的狀況，做個實際操作。

刪除欄位

以我們使用的 mytest 資料庫爲例，t2 資料表中存有學生的資料，所以在 t1 資料表中的 name 欄位就是多餘的，因此，我們可以將欄位刪除：

```
mysql> ALTER TABLE t1 DROP name;      '刪除了 t1 資料表中的 name 欄位
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

新增欄位

我們將 t2 學生資料表中加入一個 email 的欄位，參考底下的範例。我們新增的欄位名稱是 email，型別是 char(30)，使用『AFTER』關鍵字將新增的欄位放在 address 欄位的後面。

```
mysql> ALTER TABLE t2 ADD email char(30) AFTER address;
Query OK, 5 rows affected (0.00 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

新增欄位時，可以使用『AFTER 欄名』設定置於某個欄位後面，另外也可以使用『FIRST』關鍵字，將新增的欄位放在整個資料表的最前面。
若使用『FIRST』關鍵字的話，是指定將新增的欄位置於整個資料表的最前面，所以在『FIRST』關鍵字後面，不用特別指定要放在那一個欄位之前，實際上，如果使用『FIRST』又加上欄位名稱的話，反正會出現錯誤訊息。
底下查詢的結果，可以知道多了一個 email 的欄位。

```
mysql> select * from t2;
+------+------+--------+-------------+-------+
| no   | name | phone  | address     | email |
+------+------+--------+-------------+-------+
|    1 | 陳一 | 123456 | 台東市中華路 | NULL  |
|    2 | 孫二 | 234567 | 台東市大同路 | NULL  |
|    3 | 張三 | 345678 | 台東市中正路 | NULL  |
|    4 | 李四 | 456789 | 台東市新生路 | NULL  |
|    5 | 王五 | 567890 | 台東市更生路 | NULL  |
+------+------+--------+-------------+-------+
5 rows in set (0.01 sec)
```

修改欄位

欄位的名稱、儲存資料的型別也可以修改，底下的例子是將我們剛才建立的 email 欄位名稱改爲 webaddress，同時也修改它的型別是 char(32)：

```
mysql> ALTER TABLE t2 CHANGE email webaddress char(32);
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

再做一次查詢，可以看到欄位名稱已經改掉了。

```
mysql> select * from t2;
+------+------+--------+-------------+------------+
| no   | name | phone  | address     | webaddress |
+------+------+--------+-------------+------------+
```

```
|    1 | 陳一 | 123456 | 台東市中華路 | NULL       |
|    2 | 孫二 | 234567 | 台東市大同路 | NULL       |
|    3 | 張三 | 345678 | 台東市中正路 | NULL       |
|    4 | 李四 | 456789 | 台東市新生路 | NULL       |
|    5 | 王五 | 567890 | 台東市更生路 | NULL       |
+------+------+--------+--------------+------------+
5 rows in set (0.00 sec)
```

第七項　資料表最佳化

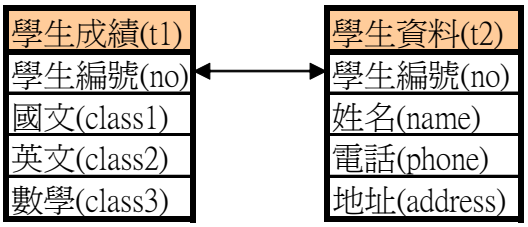
資料表在經過多次的 delete 指令刪除記錄、或 update 指令更新記錄，往往會使得資料表中有部份是沒有使用的空間，就像硬碟檔案存取多次後會使不連續區段增加，而影響到 MySQL 的效果，就必須使用『optimize table』指令來改善這個狀況：

```
mysql> OPTIMIZE TABLE t1;
+-----------+----------+----------+----------+
| Table     | Op       | Msg_type | Msg_text |
+-----------+----------+----------+----------+
| mytest.t1 | optimize | status   | OK       |
+-----------+----------+----------+----------+
1 row in set (0.01 sec)
```

第六章　資料庫應用

第一節　關聯式資料庫

關聯式資料庫，才是資料庫好玩、而且功能強大的地方，在一般的 SQL 資料庫中都可以建立多個資料表之間的關聯，舉個例子來實地操作：



這裡有二個資料表的關聯圖，左方的資料表就是我們之前所做的練習，你應該可以知道『學生成績(t1)』表示資料表的名稱，我加上底色讓它和其它的欄位資料有所區別。接下來的學生編號(no)、國文(class1)、英文(class2)、數學(class3)都是資料表中的欄位。
右方的資料表是我們要新增的，你應該可以知道資料表、各個欄位的名稱，接下來我們來建立這個資料表：

```
mysql> create table t2 (                ‘建立資料表 t2
    -> no integer,                      ‘學生編號，整數
    -> name char(8),                    ‘姓名，8 個字元
    -> phone char(16),                  ‘電話，16 個字元
    -> address char(60)                 ‘地址，60 個字元
    -> );
Query OK, 0 rows affected (0.00 sec)
```

接下來依序輸入每個學生的資料：

```
mysql> insert into t2 values (1,'陳一','123456','台東市中華路');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t2 values (2,'孫二','234567','台東市大同路');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t2 values (3,'張三','345678','台東市中正路');
Query OK, 1 row affected (0.00 sec)

mysql> insert into t2 values (4,'李四','456789','台東市新生路');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into t2 values (5,'王五','567890','台東市更生路');
Query OK, 1 row affected (0.00 sec)
```

取得二個資料表的內容

前面提到的資料表的關聯，主要是可以透過二個或二個以上的資料之間的關聯，以取得另一個資料表的資料。二個資料表關聯的欄位，我們稱爲 Foreign Key，只是，在 MySQL 中，我們沒有辦法使用 Foreign Key 來將二個資料表做關聯。不要難過，這是有原因的，使用 Foreign Key 雖然可以維護資料表間的正確性（在 Access 中叫『強迫參考完整性』），但是卻會影響資料庫查詢、刪除、新增資料的速度，而 MySQL 的優點就在於快速、穩定，而且，Foreign Key 的動作實際上交由我們來處理，也是一件容易的事，來看看底下的例子：

```
mysql> select * from t1,t2 where t1.no=t2.no;
+------+------+--------+--------+--------+------+------+--------+-------------+
| no   | name | class1 | class2 | class3 | no   | name | phone  | address     |
+------+------+--------+--------+--------+------+------+--------+-------------+
|    1 | 陳一 |     55 |     45 |     50 |    1 | 陳一 | 123456 | 台東市中華路 |
|    2 | 孫二 |     30 |     20 |     25 |    2 | 孫二 | 234567 | 台東市大同路 |
|    3 | 張三 |     70 |     85 |     68 |    3 | 張三 | 345678 | 台東市中正路 |
|    4 | 李四 |     45 |     75 |     45 |    4 | 李四 | 456789 | 台東市新生路 |
|    5 | 王五 |     95 |     90 |     86 |    5 | 王五 | 567890 | 台東市更生路 |
+------+------+--------+--------+--------+------+------+--------+-------------+
5 rows in set (0.00 sec)
```

在上面的例子，使用『select * from t1,t2 where t1.no=t2.no;』敘述，透過 MySQL 由 t1 及 t2 這二個資料表取出所有的欄位，而比較的條件『t1.no=t2.no』，是經由 t1 資料表的 no 欄位對應 t2 資料表的 no 欄位，以一筆記錄對著一筆記錄的方式，將學生的成績與學生的資料互相對照而得到我們要的資料內容。
我們也可以只取出我們要的欄位內容，例如底下的例子，在 select 後列出我們要的欄位名稱：

```
mysql> select t1.no, t2.name, t2.phone, t1.class1, t1.class2, t1.class3
    -> from t1,t2
    -> where t1.no=t2.no;
+------+------+--------+--------+--------+--------+
| no   | name | phone  | class1 | class2 | class3 |
+------+------+--------+--------+--------+--------+
|    1 | 陳一 | 123456 |     55 |     45 |     50 |
|    2 | 孫二 | 234567 |     30 |     20 |     25 |
|    3 | 張三 | 345678 |     70 |     85 |     68 |
|    4 | 李四 | 456789 |     45 |     75 |     45 |
|    5 | 王五 | 567890 |     95 |     90 |     86 |
+------+------+--------+--------+--------+--------+
5 rows in set (0.00 sec)
```

第二節　資料切割

一個完善的資料庫，是由適當分割的資料表所建立起來的，就像前面的例子，如果我們將學生的成績與學生的個人資料，分別儲存於不同的資料表中，最大的好處在於可以獨立的新增學生的成績，再使用關聯的方式取出我們要的資料。

我們實際來操作看看，底下新增的資料都是單純的學生分數而已：

```
mysql> insert into t1 values (1,null,65,55,60);     'null 表示不輸入資料
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (2,null,40,30,35);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (3,null,80,95,78);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (4,null,55,85,55);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (5,null,100,100,100);
Query OK, 1 row affected (0.00 sec)
```

透過關聯，我們可以將同一個學生的所有分數都顯示出來，例如：

```
mysql> select t1.no, t2.name, t2.phone, t1.class1, t1.class2, t1.class3
    -> from t1, t2
    -> where t1.no=t2.no ;
+------+------+--------+--------+--------+--------+
| no   | name | phone  | class1 | class2 | class3 |
+------+------+--------+--------+--------+--------+
|    1 | 陳一 | 123456 |     55 |     45 |     50 |
|    1 | 陳一 | 123456 |     65 |     55 |     60 |
|    2 | 孫二 | 234567 |     30 |     20 |     25 |
|    2 | 孫二 | 234567 |     40 |     30 |     35 |
|    3 | 張三 | 345678 |     70 |     85 |     68 |
|    3 | 張三 | 345678 |     80 |     95 |     78 |
|    4 | 李四 | 456789 |     45 |     75 |     45 |
|    4 | 李四 | 456789 |     55 |     85 |     55 |
|    5 | 王五 | 567890 |     95 |     90 |     86 |
|    5 | 王五 | 567890 |    100 |    100 |    100 |
+------+------+--------+--------+--------+--------+
10 rows in set (0.02 sec)
```

如果我們只想顯示座號爲 1 的學生成績資料，就可以這樣查詢：

```
mysql> select t1.no, t2.name, t2.phone, t1.class1, t1.class2, t1.class3
    -> from t1, t2
    -> where t1.no=t2.no and t1.no=1 ;
+------+------+--------+--------+--------+--------+
| no   | name | phone  | class1 | class2 | class3 |
+------+------+--------+--------+--------+--------+
|    1 | 陳一 | 123456 |     55 |     45 |     50 |
|    1 | 陳一 | 123456 |     65 |     55 |     60 |
+------+------+--------+--------+--------+--------+
2 rows in set (0.00 sec)
```

第一項　分割的優點

到目前爲止，我想你應該有些些明白資料表切割的重要性，說的簡單一點，資料表分割的好處在於避免資料的重覆輸入、資料的正確性、節省硬碟的空間。
以我們所做的例子而言，如果學生資料要更改，那就更改『學生資料 t2』的資料表，每次的考試資料，就通通放在『學生成績 t1』的資料表，至於查詢相關的資料時，就透過二個資料表的 no 欄位，當做資料表的關聯，就可以計算、得到很多相關資料了。

第二項　分割的方式

資料表的分割，是個挺麻煩的事，可以仔細的研讀專門的書籍，也可以記著底下二點：

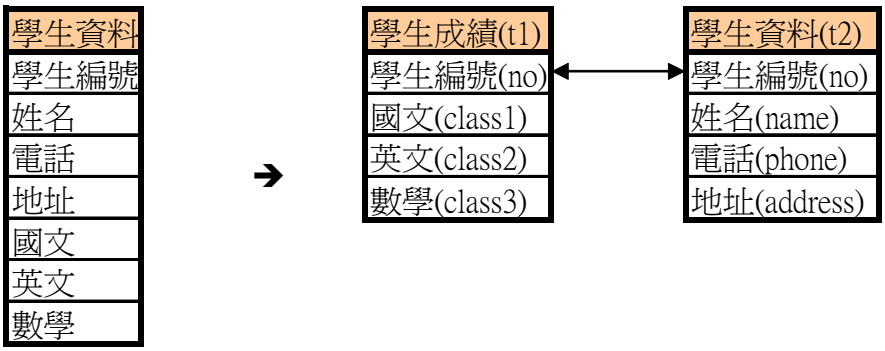
1. 欄位資料重覆：同一個欄位中所輸入的資料，總是那幾項時，就可以考慮獨立成爲一個資料表了。
2. 欄位資料的依循：不同的欄位與欄位之間，可能是有主從的關係，例如學生的電話、地址，是依著學生欄位的，而不會依著國文、英文、數學這些欄位資料；又，書籍的訂價是依著書籍本身，而不是依循日期、數量的欄位。
   如果在資料表中有這種狀況，也可以考慮將這些欄位資料獨立成爲一個資料表了。

第三項　實際例子

學生成績

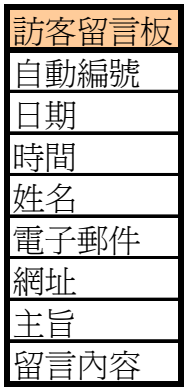
我們原本的學生成績，原本可能會規劃成左方的欄位，但是，想想看，如果每次輸入學生另一次的成績時，就得輸入學生的姓名、電話、地址，要不然就不知道

這一列的成績是那一個學生的。
這不是好辦法，因此，我們就將它更改爲右方的資料表，將和學生姓名有直接關係的電話、地址獨立成爲新的資料表，而在這個新的資料表中多留下一個『學生編號』的欄位，當成二個資料表間的關聯，因此，就得到了右方新的資料表。



留言板

再來看看留言板的資料表，應該具有的欄位如下：

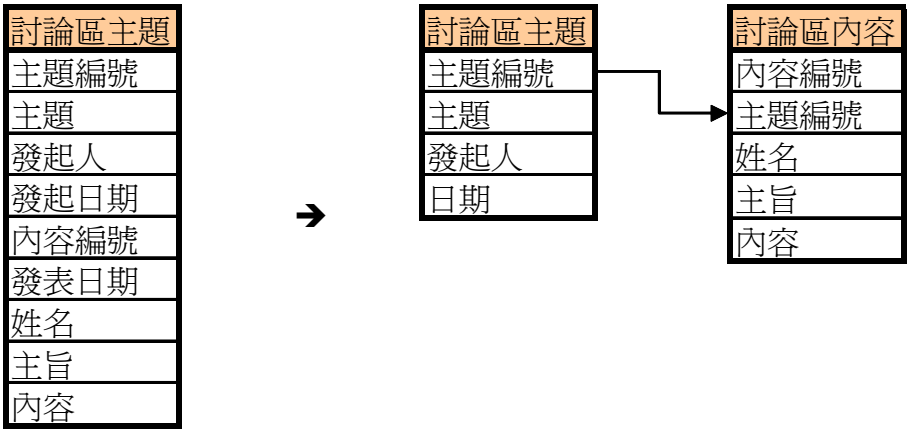


留言板部份，分割的動作就可有可無，要將資料表分割的話，可以將姓名、電子郵件、網址分割成新的資料表，但是，網路上所見到的留言板大部份都沒有分割，這是有原因的，主要是因爲留言板是公開、隨意讓訪客張貼訊息的地方，有的時候訪客（就算是熟客）會因爲某些因素（很難過、很高興、很無聊…）而輸入不同的姓名、電子郵件…等，所以，就使用一個資料表，也行！

主題討論區

左方的欄位內容是主題討論區可能會使用的資料欄位，這種資料表結構，會在訪客每一次輸入新的討論內容時，就重覆的輸入了主題、發起人、發起內容，因此，我們可以分爲二個資料表，如右圖，一個是記錄討論區主題的資料表，一個是記

錄討論區內容的資料表，在其中使用主題編號這個欄位做爲二個資料表的關聯。



第七章　MySQL 資料型別

第一節　數值

在數值型別部份，可以加上『[UNSIGNED] [ZEROFILL]』
UNSIGNED 的設定是定義資料型別都是正的數值，沒有加上的話，表示數值的範圍包含負值。
若加上 ZEROFILL 設定，則 MySQL 同時也會加上 UNSIGNED 的設定。

型別	容量
TINYINT	1 byte 數值由-128 ～ 127，或 0 ～ 255
SMALLINT	2 bytes 數值由-32768 ～ 32767 或 0 ～ 65535
MEDIUMINT	3 bytes 數值由-8388608 ～ 8388607 或 0 ～ 16777215
INT INTEGER	4 bytes 數值由-2147483648 ～ 2147483647 或 0 ～ 4294967295
BIGINT	8 bytes 由-9223372036854775808 ～ 9223372036854775807 或 0 ～ 18446744073709551615
FLOAT	4 bytes
DOUBLE	8 bytes
REAL	8 bytes

第二節　日期時間

型別	容量
DATE	3 bytes 由 '1000-01-01' ～ '9999-12-31'
DATETIME	8 bytes 由 '1000-01-01 00:00:00' ～ '9999-12-31 23:59:59'
TIMESTAMP	4 bytes 由'1970-01-01 00:00:00'到 2037 年 顯示的格示爲 YYYYMMDDHHMMSS
TIME	3 bytes

	時間由'-838:59:59' ～ '838:59:59'
YEAR	1 byte

第三節　字串

型別	容量
CHAR(M)	M bytes，M=1 ～ 255，最大 255 字元
VARCHAR(M)	L+1 bytes，最大 255 字元，可變長度的型別
TINYBLOB TINYTEXT	L+1 bytes，最大 255 字元
BLOB TEXT	L+2 bytes，最大 65536 字元
MEDIUMBLOB MEDIUMTEXT	L+3 bytes，最大 16777215 字元
LONGBLOB LONGTEXT	L+4 bytes，最大 4294967295 字元

第四節　其它

另外還有二個關鍵字，auto_increment 代表的是自動增加的欄位，我們不需要給它任何資料，它會在每新增一個記錄時，自動增加這個欄位的值；primary key 指的是主索引，也就是系統預設以這個欄位爲排序依據的主要欄位。
這二種型式，大多數都是附加在 integer 型別之中，例如：

```
prikey integer auto_increment primary key
```

如此一來，這個欄位的資料就會是這個資料表中唯一、不重覆的資料，很適合我們用於更新資料、或者是刪除資料。

第八章　在 PHP 中使用 MySQL

在 PHP 中有幾個函數可以存取 MySQL 資料庫，介紹如下，至於其中使用的敘述，和 MySQL 中使用的方式相同。

第一節　連接資料庫

連接資料庫的函數，就像使用『mysql –q』方式，如下：

```
mysql_connect (主機, 帳號, 密碼);
mysql_connect (“localhost”, “root”, “1234”);
```

第二節　建立、刪除、使用資料庫

就像是使用『create database mytest』、『drop database mytest』、『use mytest』的指令一樣，如下：

```
mysql_create_db("mytest");          ‘建立 mytest 資料庫
mysql_drop("mytest");               ‘刪除 mytest 資料庫
mysql_select_db("mytest");          ‘使用 mytest 資料庫
```

第三節　查詢

這是最常用到的部份，都是使用『mysql_query(“查詢敘述”)』的函數，來看個實際例子比較容易了解：

建立資料表

```
$sql="Create Table t1 (no integer, name char(8), phone char(16), address
char(60))";
mysql_query($sql);
```

查詢資料

```
$sql="select * from t1";
$rows=mysql_query($sql);
```

第四節　mysql_query 的傳回值

使用 mysql_query()函數傳回來的資料型式是一個指標，必須使用以下的函數來將指標所指向的記錄讀取出來：

第一項　mysql_fetch_row( )　讀取資料

可以配合 list()函數將資料讀至每一個變數之中：

```
list($no, $name, $class1, $class2, $class3)=mysql_fetch_row($rows)
```

使用 list( )函數讀取資料，會自動將指標移到下一筆記錄中，方便下一次資料的讀取。

第二項　mysql_num_rows( )　讀取總共有多少列

mysql_query 的傳回值是陣列型式，由 mysql_num_rows( )函數來取得總共有多少筆記錄：

```
$num=mysql_num_rows($rows);
```

第三項　mysql_data_seek( )　變更讀取位置

如果資料不需逐筆方式讀取，那就可以使用 mysql_data_seek( )函數來變更它的位置：

```
mysql_data_seek($rows, 3);
```

第九章　實作練習--訪客留言板

資料表名稱－guestbook
相關網頁－所有以 guest 開頭的網頁

Guestcreate.php － 建立 gb 資料表

```
<?php
    mysql_connect("localhost","root","1234");  connect 是連結的意思，故這行是連結到 mysql 資料庫，
                                    localhost 爲本地端主機，root 爲登入 mysql 的使用者名稱，1234 爲其密碼

    mysql_create_db("mydb");          create 爲建立的意思，db 爲 database（資料庫）的縮寫，
                                    故這行是建立新資料庫，建立的名稱爲 mydb

    mysql_select_db("mydb");          select 爲選擇，故這行爲選擇 mydb 資料庫做爲底下使用的預設資料庫，
                                    也就是說，以後所有和 mysql 有關的動作，就是針對 mydb 這個資料庫而言。
                                    但是，如果是另一個網頁，就和這裡選擇使用的 mydb 資料庫不一定有關係，
                                    通常每一個 php 網頁中要自行指定要選擇使用那一個資料庫使用。

    $sql="create table guestbook (    table 在資料庫中爲資料表的意思，故這行是建立一個資料表，名爲 guestbook
        gbprikey integer auto_increment primary key,      欄位 gbprikey，整數型別，自動增加，而且做爲 guestbook 的主要鍵
                                    自動增加，是指我們增加一筆記錄時，該欄位會自動加一，所以這欄位會成爲
                                    這個資料表的主要鍵，主要鍵有一個特性，就是『不可重覆』，即然會自動增加，
                                    也就代表它不會重覆。也因爲它會自動增加，所以在新增記錄時，這一欄我們給
```

```
                                null 值，也就是不給它任何資料，它會自動增加
        gbdate datetime,        欄位 gbdate，型別是日期時間，儲存發送留言的時間
        gbname char(60),        欄位 gbname，型別是 60 字元，儲存姓名
        gbemail char(60),       欄位 gbemail，型別是 60 字元，儲存電子郵件
        gbwebaddress char(60),  欄位 gbwebaddress，型別是 60 字元，儲存網址
        gbsubject char(255),    欄位 gbsubject，型別是 255 字元，儲存主旨
        gbcontent text )";      欄位 gbcontent，型別是 text，最大可到 64KB，儲存所有留言
    mysql_query($sql);          執行$sql 的字串，也就是真正的產生 guestbook 資料表，及其中的欄位格式
?>
```

connectdb.php — 連結資料庫

這個檔案的作用，是要嵌入到 index.php 檔案，要和 mysql 資料庫連結。將它獨立出來，可以移植到其它電腦時會比較方便修改。

```
<?
    mysql_connect("localhost","root","123456");連結到 mysql 資料庫
    mysql_select_db("mydb");        選擇使用 mydb 資料庫
?>
```

index.php — 首頁，其它網頁使用 include 的方式藉入至 index.php 網頁之中

在 index.php 這個網頁中，主要透過表格方式分爲左方功能表區，及右方的顯示內容區。而變數$Act 的作用在於決定 index.php 要嵌入那一個檔案到右方的內容區，實際上也就是顯示不同網頁內容的決定因素。我們指定$Act 數值小於 50 的，是訪客都可以點選顯示的內容，而大於 51 的，就是只有系統管理可以使用的網頁內容。

```
<?
    include("connectdb.php");       由於將這個檔案嵌入到 index.php 中，所以在整個 index.php 中，都是有連結
                                    到 mysql，以及選擇使用 mydb 這個資料庫的狀態，往後就不需再額外設定了
```

```
    //判斷是否登入過系統管理員的動作
    if ((!($adminlogin)) and ($Act>=51)) {如果有登入過系統管理員的話，就會產生$adminlogin 的變數，所以從是否
                                        有這個變數可以很簡單的判斷使用者有沒有登入通過系統管理員
                                        而且，也只有通過系統管理員的狀態，才允許使用$Act 大於 51 的其它功能，
                                        例如刪除留言的功能就設定爲$Act 大於 51
            header("location:index.php"); 如果沒有通過系統管理員的登入，則重新導向回到 index.php 網頁，
                                        這個動作主要是要去除$Act 的變數，不讓使用者使用只有系統管理員才能用的
                                        功能，重新回到 index.php 的單純首頁
    }
?>
<html>
<head>
    <title>網頁標題</title>            網頁標題輸入在這裡，這裡的內容會顯示在標題列上面
    <link rel=stylesheet href="style<?php echo rand(1,3); ?>.css">
                                        rand(1,3)會亂數產生數字 1,2,3，這一行的作用在於亂數使用 style1.css,
                                        style2.css, style3.css 這三個樣式檔案，檔案當然要存在
</head>

<body>                                  網頁內容開始

<table width=700 border=0 align=center>    網頁中最主要的表格，劃分爲二欄，左方放置功能表，右方顯示內容
    <tr><td colspan=2>                  表格中的第一列，合併左右二欄，顯示網頁大標題
        <h1>
        <?
            //*****************************************************
            //標題列
            switch ($Act) {             既然$Act 是用來決定要嵌入顯示那一個網頁，那我們也可以用來顯示正確的
                                        網頁大標題。
```

```
                                Switch 的作用是讓程式流程直接進入到$Act 所代表的數值去執行其中的程式
                                Switch 是從{符號開始，到}符號結束
            default:            $Act 沒有任何數值、或者是沒有任何符合的條件時，進入到這個流程
                                由於這裡沒有指定 break，所以流程會進入下一個 case 繼續執行
            case 1:             當$Act 等於 1 時，進入這個流程
                echo "山賊留言板";     echo 是顯示內容到網頁上，在這裡會變成網頁的大標題
                break;          這個流程執行完畢，跳出 switch
            case 2:
                echo "新增留言";
                break;
            case 50:
                echo "管理員登入";
                break;
            case 51:
                echo "刪除留言";
                break;
        }
    ?>
    </h1>
</td></tr>                      第一列結束
<tr>                            第二列開始，分爲左右二方，各有不同作用
    <?
        //*****************************************************
        //功能表列
    ?>
    <td width=120 valign=top>   在左方這一欄，再使用表格建立功能表，美化顯示的內容
        <table width=120 border=0>
            <tr><td class=td9>    <a href=index.php?Act=1>觀看留言</a>    </td></tr>
            <tr><td class=td9>    <a href=index.php?Act=2>新增留言</a>    </td></tr>
```

```
                                透過 index.php?Act=1 這樣的方式將$Act 變數的值再傳送到 index.php 網頁中
    <?
        if (!($adminlogin)) { !符號是反相，這行代表『沒有這個變數時』，也就是說
                                若沒有登入成系統管理員時，就符合條件，echo 顯示這行內容
            echo "<tr><td class=td9>    <a href=index.php?Act=50>管理員登入</a>    </td></tr>";
        } else {                除條件之外的狀況就進入這個流程，也就是說，『有這個變數』，
                                也表示爲『有登入過系統管理員』，就可以顯示底下的刪除留言功能
            echo "<tr><td class=td9>    <a href=index.php?Act=51>刪除留言</a>    </td></tr>";
        }
    ?>
    </table>
</td>
<td>

<?
    //*********************************************************
    //內容區                    顯示在網頁右方的網頁內容
?>
    <?
        if ($Act==1) {          當$Act=1 時，嵌入 guestbook.php 的網頁內容，顯示留言內容
            include("guestbook.php");
        } elseif ($Act==2) {  當$Act=1 時，顯示新增留言的表單
            include("guestadd.php");
        } elseif ($Act==50) { 當$Act=50 時，顯示管理員登入的表單
            include("guestadminlogin.php");
        } elseif ($Act==51) { 當$Act=51 時，顯示刪除留言的網頁內容
            include("guestdel.php");
        } else {
            include("guestbook.php");  若沒有$Act 變數或不符合上列任一項，則顯示留言內容
```

```
                }

            ?>

        </td>
    </tr>
</table>

</body>
</html>
```

guestadd.php － 新增留言

加入留言內容的表單，訪客所輸入的內容會傳送到 guestadd2.php 網頁中，由該網頁處理。

```
<form method=post action=guestadd2.php>   指定接收的資料的網頁
<table border=1 width=600 align=center>
    <tr><td>姓名*    </td><td>    <input type=text name=gbname>    </td></tr>
    <tr><td>Email    </td><td>    <input type=text name=gbemail>    </td></tr>
    <tr><td>網址    </td><td>    <input type=text name=gbwebaddress>    </td></tr>
    <tr><td>主旨*    </td><td>    <input type=text name=gbsubject>    </td></tr>
    <tr><td>內容*    </td><td>    <textarea name=gbcontent rows=6 cols=50></textarea>    </td></tr>
    <tr><td>     </td><td>    <input type=submit value=傳送>    </td></tr>
</table>
</form>
```

guestadd2.php － 將留言寫入資料庫

將由 guestadd.php 中傳送來的資料內容寫入到資料庫，再重新導向回到 index.php 顯示留言內容。
在網頁一開始，先判斷是否有輸入必要的欄位，若有輸入，才真正做新增記錄到資料庫的動作。

```
<?
    if ((!($gbname)) or (!($gbsubject)) or (!($gbcontent))) { 判斷 姓名、主旨、內容 三欄位是否都有輸入資料，這裡加上!做反相測試
                                    所以若其中有一個沒有輸入，就進入這個流程，顯示訊息說明
        echo "
            請輸入 姓名、主旨、內容 的欄位資料!
            <input type=button value=回上一頁 onclick=history.back()>
                ";
    } else {                        若三欄位都有輸入資料，則進入這個流程

        include("connectdb.php");     連結、選擇使用 mydb 資料表

        $dt=date("Y-m-d H:i:s");      取得系統的日期時間
        $gbcontent=nl2br($gbcontent); 將　內容　資料中的 Enter 取代爲<br>標籤，nl2br 是 php 本身包含的函數，
                                    可記成 New Line TO <BR>

        $sql="insert into guestbook values(null,'$dt','$gbname','$gbemail','$gbwebaddress','$gbsubject','$gbcontent')";
                                    將新增記錄的敘述，放在$sql 變數之中，順序依 guestbook 資料表建立的
                                    順序排列，而第一個欄位，由於是自動增加，故使用 null 值即可，
                                    注意在一般的變數的前後有加上'單引號。
        mysql_query($sql);          將$sql 的內容真正加入到 guestbook 的資料表
                                    mysql_query()函數會真正連結到 mysql 資料庫，做查詢、新增、刪除...的動作

        header("location:index.php?Act=1");     加入完畢，將網頁內容重新導向回到 index.php，$Act=1 代表要顯示留言內容
    }
```

```
?>
```

guestbook.php — 顯示留言內容

透過分頁的方式，每一頁只顯示部份的留言筆數。

```
<?
    $rd=10;                             設定每一頁的顯示留言筆數爲 10 筆記錄
    if ($page<1) {                      $page 是用來設定目前顯示第幾頁的留言內容，如果目前頁數小於 1，
                                        則重設爲 1，這是爲了防止訪客故意設定成負數而造成錯誤
        $page=1;
    }
    $pages=($page-1)*$rd;               計算出這一頁要從那一筆記錄開始取出，例如我們看的是第二頁($page=2)，
                                        則(2-1)*10=10，表示第二頁要從第 10 筆開始取出資料，第一頁則從第 0 筆開始

    echo "                              顯示關鍵字的表單
        <form method=post action=index.php?Act=$Act>     設定 index.php?Act=$Act，在顯示成網頁時，爲成爲 index.php?Act=1
                                        因爲 index.php 會嵌入 guestbook.php 網頁，就是因爲$Act=1 的狀態，
                                        這樣設定後，在送出關鍵字查詢後，仍然會讓 index.php 嵌入 guestbook.php
        <p align=center>關鍵字：<input type=text name=kwguest value=$kwguest>      <input type=submit value=搜尋></p>
                                        指定關鍵字的變數爲 kwguest，意思爲 key word guest 的縮寫
                                        設定 value=$kwguest，在第一次顯示 guestbook.php（或未輸入關鍵字時），
                                        $kwguest 會是空值，所以沒有任何作用，但是，當輸入關鍵字之後
                                        （例如『今天』），則會變成『value=今天』，可以在網頁中繼續保留關鍵字
        </form>
        ";

    $sql="select * from guestbook where (gbname like '%$kwguest%' or gbsubject like '%$kwguest%' or gbcontent like '%$kwguest%')";
                                        查詢符合關鍵字的記錄有幾筆，這裡查詢的部份針對 gbname, gbsubject,
```

```
                                    gbcontent 三者查詢，在$kwguest 前後加上%符號，是設定關鍵字不論在該欄
                                    的前、後、中央任何位置，都會符合查詢的條件，如果並未輸入關鍵字，
                                    則$kwguest 會是空值，查詢條件會變成 gbcontent like '%%'，表示任何字元
                                    都符合查詢的結果，亦即是不作任何篩選查詢的動作
    $rows=mysql_query($sql);        真正做查詢的動作
    $allrd=mysql_num_rows($rows);   mysql_num_rows()函數會計算查詢的記錄總共有多少筆數

    echo "<p style='text-align:center;color:green;'>目前總共有 $allrd 筆記錄";
                                    顯示內容到網頁上

    $sql="select * from guestbook where (gbname like '%$kwguest%' or gbsubject like '%$kwguest%' or gbcontent like '%$kwguest%') order by
gbprikey desc limit $pages,$rd";
                                    查詢我們要的該頁留言內容，
                                    order by gbprikey desc，會以 gbprikey 做遞減排序，新的留言在前面
                                    limit $pages,$rd，若以$page=3 爲例，則$pages=(3-1)*10=20，會等於
                                    limit 20,10，代表從第 20 筆記錄，開始取出 10 筆記錄
    $rows=mysql_query($sql);        將查詢的結果存放在$rows 變數中

    while(list($gbprikey,$gbdate,$gbname,$gbemail,$gbwebaddress,$gbsubject,$gbcontent)=mysql_fetch_row($rows)) {
                                    利用 while 迴圈，條件成立時執行，以重覆顯示資料內容，
                                    mysql_fetch_row 會從$rows 擷取一筆記錄，並將指標移到下一筆記錄
                                    list 是將取出的內容依序放到其中的變數之中
                                    這個迴圈會在$rows 內容全部擷取完畢之後，無法再擷取資料時產生 false 而停止
        echo "<table width=540 border=0 align=center>
            <tr><td class=tdll>    $gbdate                " ;      第一列顯示日期時間

        if ($gbemail) {             如果有 gbemail 內容時，就會在姓名處加上電子郵件的超連結
            echo "<a href=mailto:$gbemail>$gbname</a>";
        } else {
```

```
            echo "$gbname";              如果沒有 gbemail 時，就直接顯示姓名
        }

        echo  "</td></tr>
            <tr><td>    <b>$gbsubject    </b><br>    $gbcontent    </td></tr>
                                        顯示主旨，主旨使用<b>...</b>標籤設爲粗體
                                        用<br>換行後顯示留言內容
            </table><p>";
    }                                   到這裡，留言內容會重覆顯示，直到$rows 的筆數完畢爲止

    $p1=$page-1;                        設定『上一頁』的頁數爲目前頁數減 1
    if ($p1<1) { $p1=1; }               若上一頁小於 1，則設爲 1

    $p2=$page+1;                        設定『下一頁』的頁數爲目頁數加 1
    $pageall=intval($allrd/$rd)+1;      計算總頁數爲何，$allrd 爲前面取得的總筆數，除$rd 可得總頁數
                                        intval()函數會無條件捨去小數位數，故再加 1
    if ($p2>$pageall) { $p2=$pageall; }    若下一頁的頁數大於總頁數，則以總頁數爲下一頁的值

    echo "<p><a href=index.php?Act=$Act&page=$p1>上一頁</a> / <a href=index.php?Act=$Act&page=$p2>下一頁</a>";
                                        顯示上一頁及下一頁的超連結
                                        因爲$Act=1 所以顯示留言內容，會透過 Act=$Act 達成
                                        而 page 頁數是$p1 或是$p2，則依點選上一頁或下一頁而決定
?>
```

guestadminlogin.php — 系統管理員的登入表單

```
<form method=post action=guestadmincheck.php>
<table width=400 align=center border=1>
```

```
    <tr><td>    管理員帳號    </td><td>    <input type=text name=adminname>    </td></tr>
    <tr><td>    密碼        </td><td>    <input type=password name=adminpw>    </td></tr>
    <tr><td>             </td><td>    <input type=submit value=登入>    </td></tr>
</table>
</form>
```

guestadmincheck.php — 確認登入的帳號密碼是否正確

以最基本（固定帳號及密碼）的方式來判斷管理員的帳號及密碼是否正確，實務上仍然以資料庫的方式比較洽當。

```
<?php

    if (($adminname=="admin") and ($adminpw=="1234")) {  若帳號密碼符合，則進入以下流程
        setcookie("adminlogin",$adminname);     設定 cookie，變數名稱爲$adminlogin，也是判斷是否登入管理員的依據
        header("location:index.php"); 重新導向回 index.php 首頁，但已經可以使用刪除留言的功能
    } else {                          若帳號密碼不符合，則重新回到管理員登入畫面
        header("location:index.php?Act=50");
    }

?>
```

guestdel.php — 刪除留言

這個網頁只顯示留言內容，每個留言都有一個 checkbox 可以勾選，真正刪除的動作要由 guestdel2.php 來執行

```
<?php

    $sql="select * from guestbook order by gbprikey desc";    依 gbprikey 遞減排序，取得所有的留言筆數
```

```
    $rows=mysql_query($sql);

    echo "<form method=post action=guestdel2.php>"; 指定要由 guestdel2.php 接收處理後來的動作
    echo "<table width=600 border=1 align=center>";
    echo "<input type=submit value=刪除>";
    while(list($gbprikey,$gbdate,$gbname,$gbemail,$gbwebaddress,$gbsubject,$gbcontent)=mysql_fetch_row($rows)) {
        echo "
            <tr>
                <td>    <input type=checkbox name=delguest[] value=$gbprikey>    </td>
                                設定 delguest[]陣列，若有勾選，則將 value 所代表的$gbprikey 值代入
                                若勾選三個，會變成 delguest[0], delguest[1], delguest[2]
                                其中的值，就是所勾選的$gbprikey 的值
                <td>    $gbprikey    </td>以下顯示其它相關資料
                <td>    $gbdate    </td>
                <td>    $gbname    </td>
                <td>    $gbsubject    </td>
                <td>    $gbcontent    </td>
            </tr>
        ";
    }
    echo "</table>";
    echo "</form>";

?>
```

guestdel2.php — 刪除留言，從資料庫中確實刪除

接收從 guestdel.php 送來的資料，將欲刪除的內容真正從 guestbook 中刪除。

```
<?php

    include("connectdb.php");          連結資料庫

    $num=count($delguest);             取得勾選的數量
                                       count()函數會計算陣列的項目有幾個，所以可以取得勾選欲刪除的留言項目
                                       注意其中的陣列並未加上[]陣列符號，加上了反而會有錯誤

    for($i=0;$i<$num;$i=$i+1) {        若只勾選三個，則迴圈只執行三次
        $sql="delete from guestbook where gbprikey='$delguest[$i]'";
                                       依序刪除該筆記錄
        mysql_query($sql);
    }
    header("location:index.php?Act=51");   繼續回到刪除留言 guestdel.php 的內容
?>
```

呼～～　大功告成！

第十章　PHP 函數

第一節　和 MySQL 相關的函數

第一項　mysql_connect( )　連接資料庫

使用 PHP 連接到 MySQL 的資料庫。

```
$link=mysql_connect("localhost", "root", "1234");
```

如果連線成功，會傳回一個整數給$link，失敗的話傳回 false。

第二項　mysql_close( )　關閉 MySQL 的連線

關閉 PHP 和 MySQL 的連線。

```
$chok=mysql_close($link);
```

將$link 的連線關閉，成功的話傳回 true，失敗則傳回 false。如果沒有給函數$link，則會關閉最後的連線。另外，就算不執行 mysql_close 函數，PHP 也會在該 PHP 網頁執行完畢時自動關閉和 MySQL 的連線。

第三項　mysql_create_db( )　建立資料庫

建立新的資料庫。

```
$link=mysql_connect("localhost", "root", "1234");
$chok=mysql_create_db("mytest", $link);
```

在 PHP 連線到 MySQL 資料庫之後，可以建立新的資料庫，建立成功則傳回 true，失敗則傳回 false。

第四項　mysql_select_db( )　選擇使用資料庫

選擇使用的資料庫。

```
$chok=mysql_select_db("mytest", $link);
```

在 PHP 連線到 MySQL 資料庫之後，可以選擇使用的資料庫是那一個，成功的話

傳回 true，失敗則傳回 false。

第五項　mysql_query( )　查詢資料

對資料庫做 create table、insert、delete、update、select 的查詢動作，可以說是最常使用的函數。

```
$sql="create table t1 (name char(10), email char(30) ) ";
$chok=mysql_query($sql);              '假設 t1 資料表只有 name、email 欄位

$sql="insert into t1 values('kingbig', 'webmaster@mail.kingbig.com')";
$chok=mysql_query($sql);

$sql="update t1 set email='kingbig@kingbig.com' where name='kingbig' ";
$chok=mysql_query($sql);

$sql="delete from t1 where name='kingbig' ";
$chok=mysql_query($sql);
```

上述動作分別做了建立新資料表、新增、更新、刪除資料，執行成功的話傳回來的值都是 true，失敗則傳回 false。
如果做的是 select 查詢的動作，有一些不太相同：

```
$sql="select * from t1";
$rows=mysql_query($sql);
```

如果查詢動作失敗的話，傳回的是 false，如果查詢動作成功的話，傳回的是讀取資料的指標，我們可以透過指標來將記錄一筆一筆的方式讀出來。

第六項　mysql_num_rows( )　取得記錄的筆數

取得由 mysql_query 查詢所得到的記錄筆數。

```
$sql="select * from t1";
$rows=mysql_query($sql);
$num=mysql_num_rows($rows);
```

傳回的就是使用 select 查詢所得到的資料數量。

第七項　mysql_fetch_row( )　抓取記錄為陣列型式

將使用 mysql_query 所得到的資料指標中的該筆記錄讀取出來，並且自動將資料的指標移到下一筆記錄。

讀取出來該筆記錄會是陣列型式，以數字爲索引的陣列型式。

```
$sql="select * from t1";
$rows=mysql_query($sql);
list($name, $email)=mysql_fetch_row($rows);
```

雖然傳回的是以數字爲索引的陣列，但我的習慣都是使用 list( )函數將各個值分別存放在不同的變數之中，比較有意義，也比較容易記憶。
使用 mysql_fetch_row 函數之後，指標會自動往下移一筆，因此，我們若是重覆使用 mysql_fetch_row 函數，自然就將所有資料讀出來了，所以，這個函數也常常用在迴圈之中，以方便資料的讀取及運用。

第八項　mysql_fetch_array( )　抓取資料為陣列型式

和 mysql_fetch_row 類似，不同的地方是 mysql_fetch_array 傳回來的資料型式是以字串（也就是欄位名稱）爲索引的陣列。

```
$sql="select * from t1";
$rows=mysql_query($sql);
$data=mysql_fetch_array($rows);
echo $data['name'];
echo $data['email'];
```

取出一筆記錄之後，指標也會自動往下移動。

第九項　mysql_data_seek( )　移動指標

移動指標到某筆記錄。

```
$sql="select * from t1";
$rows=mysql_query($sql);
$chok=mysql_data_seek($rows, 3);
```

上例在將指標移到 3 的位置，需注意指標是由 0 開始，也就是說，上例實際上是將指標移到第四筆記錄。
移動指標成功傳回 true，失敗則傳回 false。

第二節　字串相關函數

第一項　stripslashes( )　中文問題的解決方法

第二項　nl2br( )　將 Enter 取代為
標籤

第三項　strtoupper( )　字串轉為大寫

第四項　strtolower( )　字串轉為小寫

第五項　trim( )　去除字串前後空白

第六項　join( )　將陣列資料結合為字串

第七項　explode( )　將字串分割為陣列型式

第八項　strlen( )　字串長度

第九項　str_replace( )　取代字串中的特定字元

第十項　substr( )　取出字串中的字串