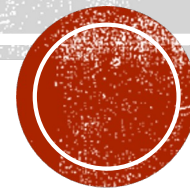


# PHP 網站設計

林新德

shinder.lin@gmail.com



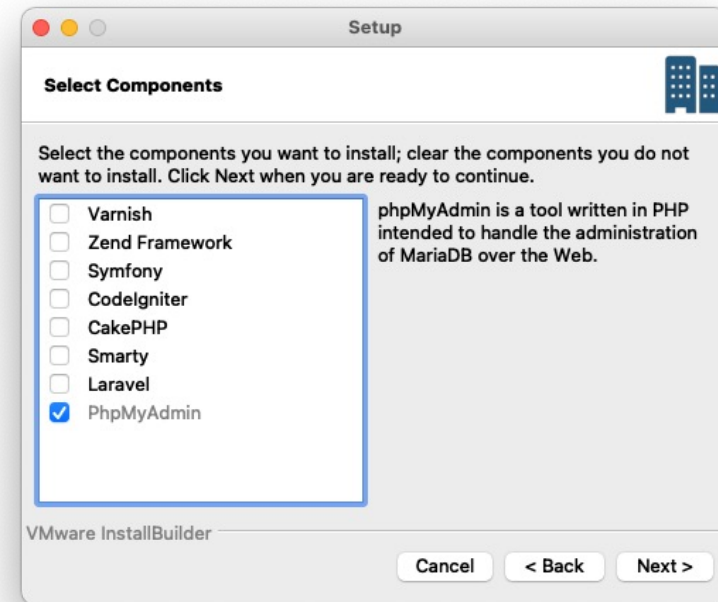
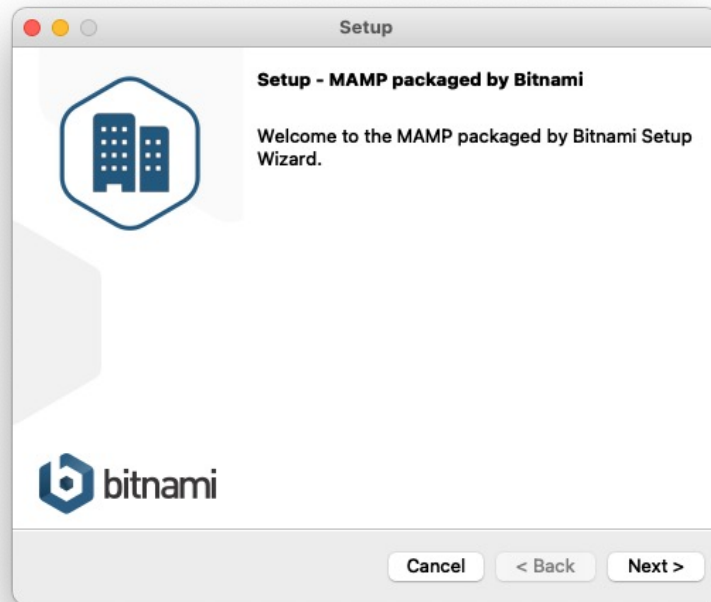
參考專案 <https://bitbucket.org/lsd0125/mfee29-php/>

# 1. 開發執行環境

- 安裝包：Apache + MySQL (或 MariaDB) + PHP (Apache module)
- Mac平台可以使用：
  - XAMPP ([https://www.apachefriends.org/zh\\_tw](https://www.apachefriends.org/zh_tw))，使用 ~160M 不是 VM 的版本
  - MAMP ( <https://www.mamp.info> )
  - Bitnami MAMP packaged 😊
- Windows平台可以使用：
  - XAMPP ([https://www.apachefriends.org/zh\\_tw](https://www.apachefriends.org/zh_tw))
  - WAMP ( <http://www.wampserver.com/en/> )
  - Bitnami MAMP packaged 😊

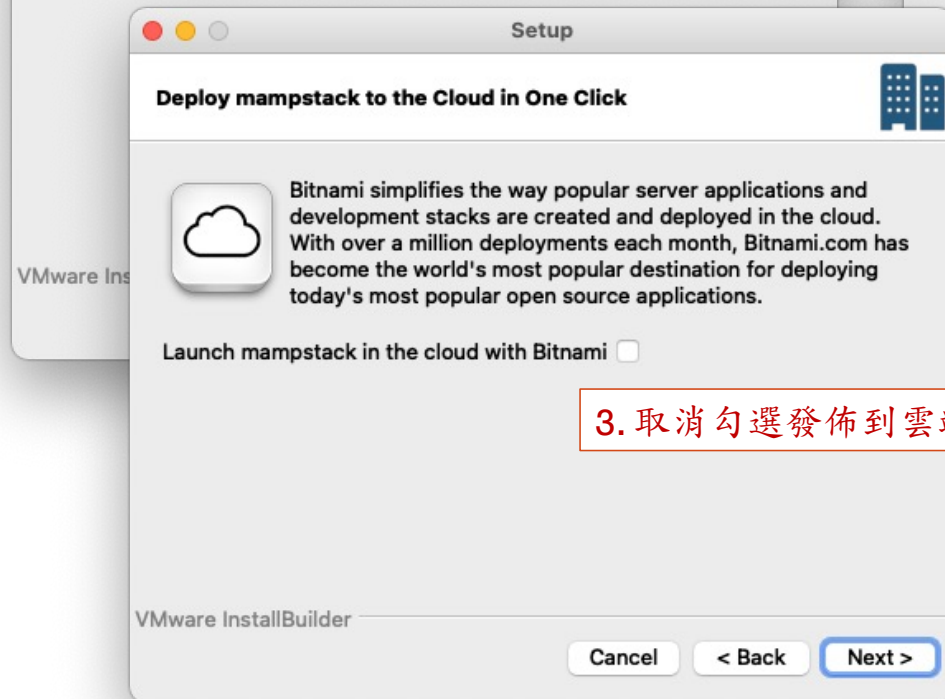
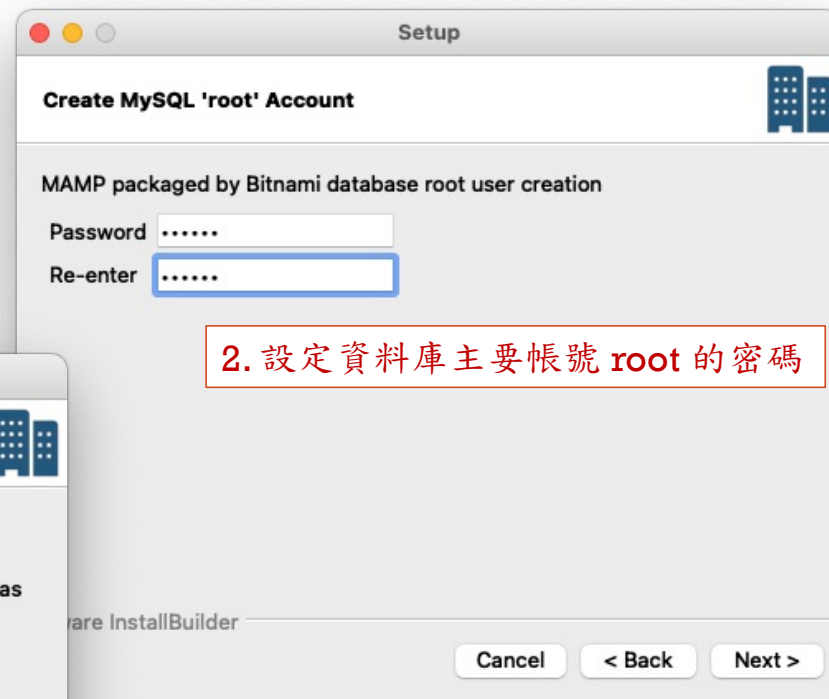


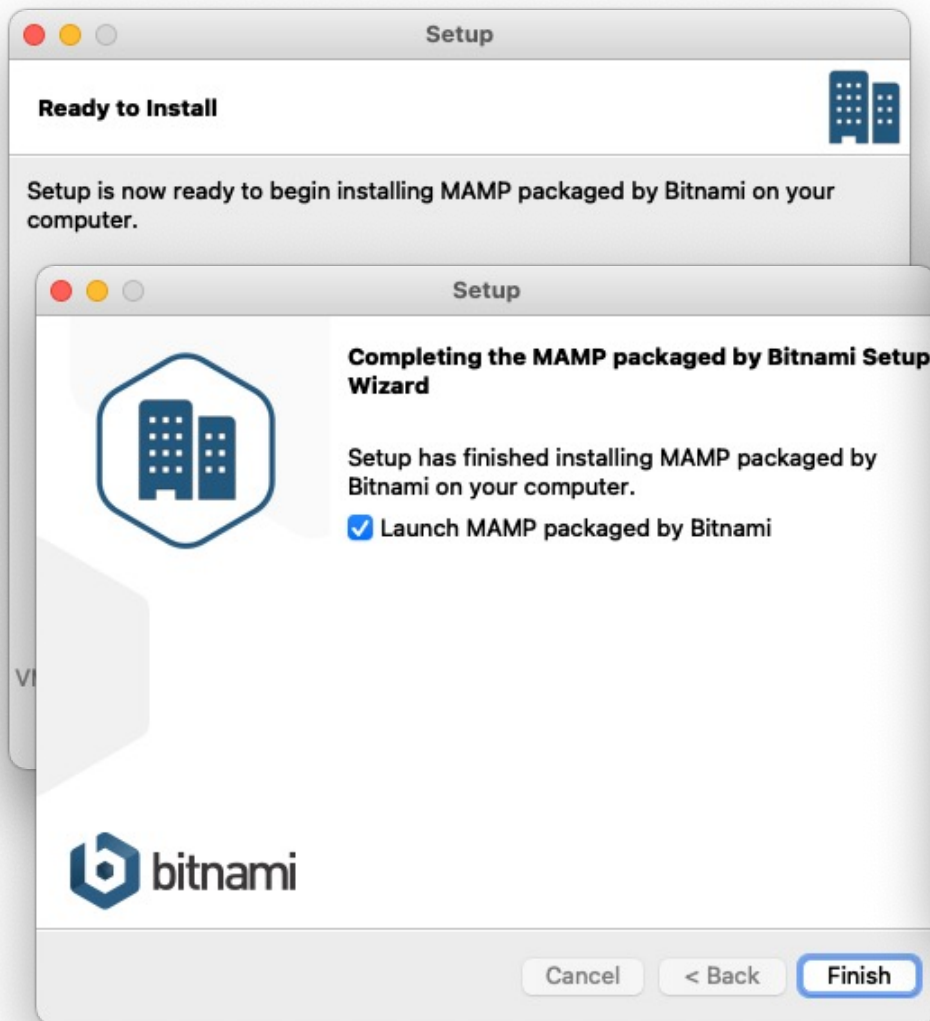
# 1.1 安裝 Bitnami MAMP packaged



\*大多的 PHP 框架可以不用安裝，有 **PhpMyAdmin** 即可







## 1.2 開啟 PhpMyAdmin 介面



◆ 啟動伺服器（**Apache** 和 **MariaDB**）後，  
到控制面板點擊「**Open phpMyAdmin**」。



## 1.3 編輯環境

- Visual Studio Code (免費軟體)
- PhpStorm (付費軟體)



## 1.4 建立專案

- 將 Document root 內的 index.html 修改檔名為 index\_.html
- Document root 位置：[/Applications/mampstack-8.0.25-0/apache2/htdocs](#)
- 在 Document root 開立一個專案的資料夾 my-proj。一般命名不使用中文或奇怪的符號。通常使用英文字母數字，dash或underscore。
- 使用 VSCode 開啟 my-proj 資料夾
- PHP程式碼的標示以 `<?php` 開頭，`?>` 結束。當 php 程式之後沒有其它 HTML 內容時，php 程式的結束標記 `?>` 可以省略。





## 1.5 第一支 PHP 程式

```
<?php
# 儲存為 info.php
phpinfo(); // 函式、方法，名稱不區分大小寫

/* 註解 */
// 註解
# 註解
```

用瀏覽器查看 `phpinfo.php` 。找出以下的設定值：

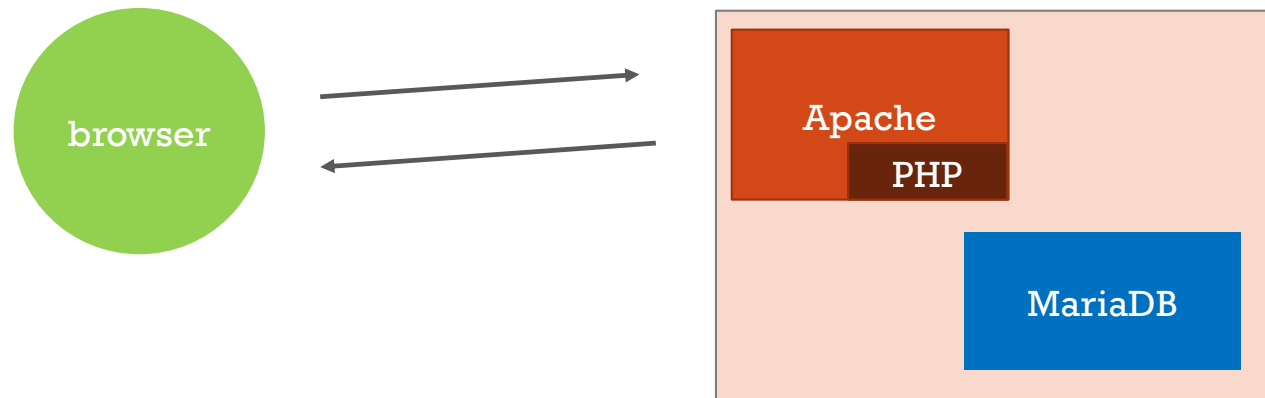
`Loaded Configuration File`

`Server Root`

`MYSQL_HOME`

`DOCUMENT_ROOT`







## 2. PHP 的運算

- PHP 一開始就是 WWW 伺服端的技術，用來撰寫動態網頁，其善長處理「文字」資料。
- HTML 內容本來就是文字，許多資料在輸出到頁面時也會自動轉換成文字。
- PHP 是 C-like 程式語言，程式裡放置空白（換行）相當有彈性，應遵守縮排規則，以利程式撰寫及除錯。
- PHP 程式的每一句敘述請標上分號（;），分號表示程式敘述的結束。只有在最後一行的程式敘述（?> 之前）可以不用標分號，其餘都應該要標示。



## 2.1 常數

- 所謂常數是不可變更的值。PHP 的常數可以分成三類：一般常數（例：17）、系統常數（例：`PHP_VERSION`）及自訂常數。
- 一般常數就是一些數值：0、-9、2.5、'abc' 等。
- 系統常數為 PHP 預設的常數，常用的有：`PHP_VERSION`（查看 PHP 版本）、`__FILE__`（PHP 檔的實體路徑）、`TRUE` 和 `FALSE`（布林值的 `true` 和 `false`）。
- 系統常數大部份可以由 `get_defined_constants()` 函式取得（取得的資料是陣列類型）。
- 自訂常數是程式開發者所定義的常數，可以使用 `define()` 函式定義。



## 2.2 自訂常數

- `define(常數名稱, 常數內容, 常數名稱是否不區分大小寫);`
- 常數名稱通常使用全部大寫的英文字，複合字時中間使用 **underscore** (`_`) 分開。
- 常數內容則可以是數值或字串。
- 常數名稱大都使用全大寫，「是否不區分大小寫」一般是不使用的。

```
<?php  
define("MY_CONSTANT", "常數 - 值在設定後不可變更");  
echo MY_CONSTANT;
```



## 2.3 變數

- PHP 的變數在使用前可以不用宣告，其為自動型別，不需要限定變數只能使用某個型別。
- PHP 變數有個特色，就是在變數前必須要有個美金符號（\$），而且變數名稱是有大小寫的區別的。
- 自訂的常數前不使用 \$，而且通常所有字母為大寫。
- 變數前必須要有 \$，有區分字母大小寫。
- PHP 變數的命名規則和一般程式語言一樣，第一個字母不可以是數字。
- 變數在設值之後，若不想再使用可用 `unset()` 刪除。



```
<?php
$my_var = 66;
$b = "22";
$c = "abc";
echo $my_var + $b; // 88
echo "<br />";
echo $my_var + $c; // 66
```





## 2.4 字串

- **PHP** 的字串標示方式使用一對雙引號 (") 或一對單引號 (')，不過兩者在使用上有所不同，雙引號標示者裡面若包含變數，會顯示變數值；單引號標示者則否。

```
<?php
$a = 'Victor';
echo "Hello, $a <br />";
echo 'Hello, $a <br />';
```

```
<?php
$a = 'Victor';
echo "Hello, $a123 <br/>";
echo "Hello, {$a}123 <br/>";
echo "Hello, ${a}123 <br/>";
echo $a. ' <br/>';
```

- 雙引號標示中，有時輸出變數內容情況會比較複雜，此時可以使用大括號解決。
- 點符號 (.) 在 **PHP** 用來做字串串接，不是使用加號 (+)，這點要特別注意。



- 雙引號和單引號在使用脫逸字元時，亦不太一樣(執行後請查看頁面原始內容)。。

```
<?php
$a = "xyz\nabc\"def\'ghi\\";
$b = 'xyz\nabc\"def\'ghi\\"';
echo $a;
echo $b;
```

```
<?php
$a = "PHP - MySQL
    是好朋友!";
$b = <<<HDOC
    <br>
    <h1>PHP - </h1>
    <div style="color:#F00;">MySQL</div>
HDOC;
echo $a;
echo $b;
```

- PHP 的字串還有個好處，就是可以換行標示，所以方便定義較長且包含換行的字串。若文字內容也包含雙引號時，可以使用「即為文件」(Heredoc)表示法。
- Heredoc 表示法是以 <<< 為開頭，緊接「標示名」(可自訂，通常為大寫字母)，加換行。Heredoc 結束時，在新的一行使用「標示名」即可。





## 3. 運算子

- PHP 的運算子和其它 C-like 的程式語言大部份是相同的，在此我們來複習一下。算術運算子除了加 + 減 - 乘 \* 除 / 四則運算外，還包括了求餘數 %。這裡的四則運算和數學的四則運算是相同的。

運算子	語法	\$a=7, \$b=2 運算結果
+	\$a + \$b	9
-	\$a - \$b	5
*	\$a * \$b	14
/	\$a / \$b	3.5
%	\$a % \$b	1



- 「`$a = $a+1`」以遞增運算子可以簡化為「`$a++`」。「`$a = $a + 3`」也有替代運算子，稱為算數指定運算子。

算數指定運算子	算數並指定	使用算數指定運算子
<code>+=</code>	<code>\$a = \$a + \$b</code>	<code>\$a += \$b</code>
<code>-=</code>	<code>\$a = \$a - \$b</code>	<code>\$a -= \$b</code>
<code>*=</code>	<code>\$a = \$a * \$b</code>	<code>\$a *= \$b</code>
<code>/=</code>	<code>\$a = \$a / \$b</code>	<code>\$a /= \$b</code>
<code>%=</code>	<code>\$a = \$a % \$b</code>	<code>\$a %= \$b</code>



- 「關係運算子」(relational operator)或稱為比較運算子。

關係運算子	說明	範例	範例結果
>	是否大於	5 > 2	true
>=	是否大於等於	5 >= 2	true
<	是否小於	5 < 2	false
<=	是否小於等於	5 <= 2	false
==	是否相等	5 == 2	false
!=	是否不相等	5 != 2	true
===	是否相等 (嚴謹)	5 === '5'	false
!==	是否不相等 (嚴謹)	5 !== '5'	true



- 條件運算子「?:」是唯一的三元運算子，其語法如下：
- 判斷值 ? 真時選擇值 : 假時選擇值
- 第一個運算元 (判斷值) 應該為布林值，或是結果為布林值的運算式。當判斷值為 **true** 時，整個運算式的結果取真時選擇值;反之，判斷值為 **false** 時，取假時選擇值。通常我們會將整個條件運算結果，存放到某個變數內，所以會如下式：
- 變數 = 判斷值 ? 真時選擇值 : 假時選擇值;



- 邏輯運算式和關係運算式有一個共同點，兩者的運算結果都是布林值，不是 **true** 就是 **false**。然而，邏輯運算式中的運算對象 (運算元)，也視為布林值處理。

邏輯運算子	意義	用法
&&	且	\$a && \$b
	或	\$a    \$b
!	非，not (優先權最高)	! \$a
and	且 (優先權比 = 低)	\$a and \$b
or	或 (優先權比 = 低)	\$a or \$b





- 運算子依性質不同分類，優先權高的類別會被先列出來，越往下優先權越低（粗略規則）：

1. 單元運算子
2. 算術運算子
3. 關係運算子
4. 邏輯運算子
5. 指定運算子

- 除了上述的運算子，分隔符號中的小括號「()」、逗號「,」也和運算式有關係。
- 小括號可以提升運算子和運算元的運算關係，以小括號包起來的運算式會優先運算。
- 逗號 可用於分隔宣告變數(或變數宣告的指定式)，並不會影響任何值。





## 4. URL 變數 ( GET 參數 )

- 網址內容裡 ? 之後為 query string 參數，也稱為 URL 變數或 GET 參數，其格式為 URL-encoded。
- PHP 可以使用 \$\_GET 預設變數取得其內容。

```
<?php  
echo $_GET['a'] + $_GET['b'];
```

- 測試網頁之後，在網址列輸入下式 ? 之後的內容。  
**http://localhost/my\_test/untitled1.php?a=12&b=24**





# 5. 流程控制

- 流程控制通常可以分成兩類：選擇敘述和迴圈敘述。
- 選擇敘述包含 if/else 和 switch/case 。
- 迴圈敘述包含 for、while、do/while、及 foreach/as 。



# 5.1 if/else

- if 敘述是選擇執行或不執行，if/else 敘述則是二選一執行。
- 以下是 if/else 選擇結構的語法：

```
if(條件式) {  
    //區塊敘述一  
} else {  
    //區塊敘述二  
}
```

- if/else 語法意義為：若條件式為 true 時，執行區塊敘述一，否則（條件式為 false 時）執行區塊敘述二。
- 視需要也可以省略 else 區塊。



- 有時 if/else 可以轉換成三元運算式。

```
<?php
$a = 0;
if( isset($_GET['a']) ){
    $a = $_GET['a'];
}
$b = isset($_GET['b']) ? intval($_GET['b']) : 0;
echo "$a, $b";
```

- else if 也可以黏在一起，寫成 elseif。



- 一般的想法是 PHP 的程式碼是嵌在 HTML 的，其實剛好是相反的，我們可以把 HTML 的內容看成是透過 echo 輸出的。

```
<?php
if (isset($_GET['name']) and $_GET['name'] == 'shinder') {
    ?>
    <p>您好! Shinder<br/>
        歡迎光臨~
    </p>
    <?php
} else {
    ?>
    <p>您好! 陌生人<br/>
        您無權使用管理系統哦!
    </p>
    <?php
}
?>
```





## 5.2 switch/case

- if/else 的巢狀結構可以解決多重選擇的問題，在這則是介紹專門設計給多選一使用的 switch 敘述。switch 並不使用條件式來決定執行的區段敘述，而是使用鍵值（通常為變數）。

```
switch (鍵值) {  
    case 條件值1:  
        // 區段1敘述  
        break;  
    case 條件值2:  
        // 區段2敘述  
        break;  
    .....  
    case 條件值N:  
        // 區段N敘述  
        break;  
    default:  
        // default區段敘述  
}
```



- 執行 switch 敘述時，會先求得鍵值，接著將鍵值和條件值 1 做比對。如果兩值相等，則執行區段 1 敘述，接著遇到 break 之後跳離 switch 敘述；如果兩值不相等，則再往下比對條件值 2，以此類推。如果鍵值和所有的條件值都不相等時，則執行「default:」下的 default 區段敘述。default 部份在 switch 敘述中是選擇性的，可有可無，不一定要放在最後面。

```
<?php
$page = isset($_GET['page']) ? $_GET['page'] : '';
switch( $page ) {
    default :
    case 'main':
        echo 'main page';
        break;
    case 'content':
        echo 'content page';
        break;
    case 'about':
        echo 'about page';
        break;
}
```



## 5.3 for 迴圈

- **起始式**—進入迴圈時，一開始執行的程式運算式，只執行一次。通常起始式，多為設定控制迴圈執行變數的起始值。
- **條件式**—判斷是否執行迴圈內區塊的依據。如果條件式的結果為 true，則繼續執行；如果為 false，則跳離迴圈。
- **步進式**—每經過一次迴圈，就會執行一次的運算式。通常步進式，多為設定控制迴圈執行變數的遞增或遞減運算式。

```
for( 起始式; 條件式; 步進式) {  
    // 區塊內敘述  
}
```



```
<table border="1">
  <tr>
    <?php
      for ($i = 0; $i < 10; $i++) {
        echo "<td> $i </td>";
      }
    ?>
  </tr>
</table>
```



## 5.4 while 迴圈

- while 迴圈和 for 迴圈很像，也可以說是 for 迴圈的簡化版，基本上 while 迴圈標頭只有用以判斷的條件式，而沒有起始式和步進式。。

```
// 起始式
while(條件式){
    // 區塊內敘述
    // 步進式
}
```

- 條件式若為 true，則執行大括弧內的區塊內敘述，執行完之後，回到前面再判斷條件式，若還是為 true 則再執行一次區段敘述，直到條件式為 false 時才跳離迴圈。
- 使用迴圈的時候都必須注意無窮迴圈，尤其是 while 迴圈。while 的控制變數並不像 for 可以直接放在敘述的標頭，所以可能會忘了加步進式，這個時候就會形成無窮迴圈。
- 條件式直接使用 true 是在「不知道要跑幾次迴圈」時使用，不過還是可以搭配跳離迴圈的關鍵字 break 來使用，讓它不至於無窮的執行下去。



## 5.5 do/while 迴圈

```
do{  
    // 區塊內敘述  
} while (條件式);
```

- do/while 結構以 do 關鍵字為開頭，接著是大括弧包起來的區塊內敘述，最後是 while 關鍵字和小括弧包著的條件式。
- 請注意，for 和 while 迴圈都是以右大括弧結尾，所以不需要再加分號(;)標示敘述的結束。但是，do/while 結尾是右小括弧，並不能代表一個區塊的結束，因此要加上分號，以標示迴圈結尾。
- 若拿 while 迴圈和 do/while 迴圈做比較，最明顯的不同就是條件式的位置。do/while 的條件式擺在結構的最後面，也就表示先執行區塊敘述一次，再做條件式判斷，這也可以說是 do/while 的特色。



## 5.6 break 與 continue

- 關鍵字 break 和 continue，可以改變迴圈的流程。break 曾經在使用 switch 敘述時看到，不過在這將介紹 break 和迴圈的關係。
- 在 switch 選擇敘述中，我們使用過 break，讓流程跳離 switch 敘述區塊。而在 for、while、do/while 迴圈中，break 的作用也是相同的，都是跳離敘述區塊（跳離迴圈的大括號範圍）。一般使用 break 是為了在特殊情況下提早跳離迴圈。
- continue 的功能是跳到迴圈的起始處。

```
<?php
for ($j = 1; $j < 7; $j++) {
    if ($j == 4)
        continue;
    echo $j;
}
```







## 6. 陣列

- 何謂陣列呢？簡單地講，陣列就是「多個擁有相同名稱的變數集合」。
- PHP 的陣列又可細分成「索引式陣列」（Indexed Array）和「關聯式陣列」（Associated Array）。
- 在PHP索引式和關聯式是可以混用的，不過依它們的特性我們會分開來討論。



# 6.1 索引式陣列

- 元素都有相同的名稱，為了區別每個陣列元素，它們都有各自的位置編號。
- 使用陣列時，傳統的方式用 `array()` 函式建立函式，也可以直接用中括號建立。
- 為了指出是哪個陣列元素時，必須以陣列名稱，後接一對中括弧 `[]`，中括弧內放入位置編號。位置編號一般也稱為索引，索引由 0 開始，接著是 1、2、3...至元素個數減一。
- 陣列的索引應該為大於或等於 0 的整數，或者結果為整數的運算式。

```
<?php
$ar = array(3, 2, 2, 0, 4, 1);
$ar2 = [3, 2, 2, 0, 4, 1];
echo $ar[1]; // 2
```



- PHP 的陣列是動態的，可以動態加入元素或移除元素，而且元素沒有限定類型。
- 用 [] 直接「建立陣列並放入第一個元素值」或「推入一個元素值」：

```
<?php  
$br[] = 3; // 相當於 array_push()  
$br[] = 2;  
$br[] = 2;  
print_r($br);
```

- `print_r()` 和 `var_dump()` 常用來查看資料內容和除錯。
- `count()` 函式能取得陣列裡元素的個數。
- PHP 的內建函式名稱，雖然大部份會用有意思的名稱，有些則用縮寫容易讓人混淆。而且由於發展的歷史，大部份的函式是全域的，而不是透過物件導向的方式歸類在某些類別下。建議初學者常查看 PHP 手冊，或在 [php.net](http://php.net) 的「search for」進行查詢。



- 若要不理會索引值是否連續，依序取得陣列的元素值，可以使用 foreach/as 迴圈。
- 列舉變數是自訂變數，由開發者自行決定。列舉變數會依迴圈執行的圈數，依序取得元素值。

```
foreach(陣列變數 as 列舉變數) {  
    //迴圈主體內容  
}
```

```
$ar = [2, 3, 12];  
$ar[9] = 7;  
foreach ($ar as $v) {  
    echo "$v <br />";  
}
```



## 6.2 關聯式陣列

- PHP的關聯式陣列是由一個字串 **key** 對應到一個 **value** 。
- 關聯式陣列可以看成是 hash table（雜湊表）的實作，只要給一個 **key**（屬性名稱）就可以得到對應的 **value**（屬性值）。
- PHP 的關聯式陣列同樣可以用 `array()` 函式或中括號建立，**key** 和 **value** 之間用胖箭頭「`=>`」來設定。「`=>`」由一個等號和大於符號組成，注意中間沒有空白。

```
<?php
$ar = [
    3      => 'abc',
    '3'    => 'def',
    'name' => 'shinder',
    'pw'   => 'pass',
];
echo "{$ar[3]} <br/>";
echo "{$ar['3']} <br/>";
echo "{$ar['name']} <br/>";
```

- ◆ 使用關聯式陣列時，通常 **key** 不使用數值，使用數值會變成使用索引式陣列。
- ◆ **key** 若可以轉換成數值，即使用雙引號標示，同樣視為相同，所以「3」和「'3'」視為相同的 **key**，**key** 是唯一的。



- 關聯式陣列也可以用 foreach/as 迴圈來取值。
- 「鍵變數」和「值變數」同樣使用「=>」指明其關係。

```
foreach(陣列變數 as 鍵變數 => 值變數) {  
    //迴圈主體內容  
}
```

```
<?php  
$ar = [  
    3 => 'abc',  
    '3' => 'def',  
    'name' => 'shinder',  
    'pw' => 'pass',  
];  
foreach ($ar as $k => $v) {  
    echo "$k : $v <br />";  
}
```



## 6.3 陣列的「=」設定

- PHP陣列在使用「=」做設定運算時，並不是設定參照（reference），而是做實體的複製。若要設定參照，必須使用 & 符號。

```
<pre><?php
    $ar = array(
        'id' => 17,
        'name' => 'shinder',
    );
    $br = $ar; // 複製一份新的
    $cr = &$ar; // 指向相同的陣列
    $ar['id'] = 26;
    print_r($ar);
    print_r($br);
    print_r($cr);
?></pre>
```



## 6.4 \$\_GET 和 \$\_POST

- PHP 用來取出客戶端傳來資料的 \$\_GET 和 \$\_POST 是以陣列的形式存在。
- 以下的例子在執行時，試著在網址後接著輸入「?a=123&bb=yes」，a 和 bb 會是 \$\_GET 的 key，而 123 和 yes 分別是對應的 value。

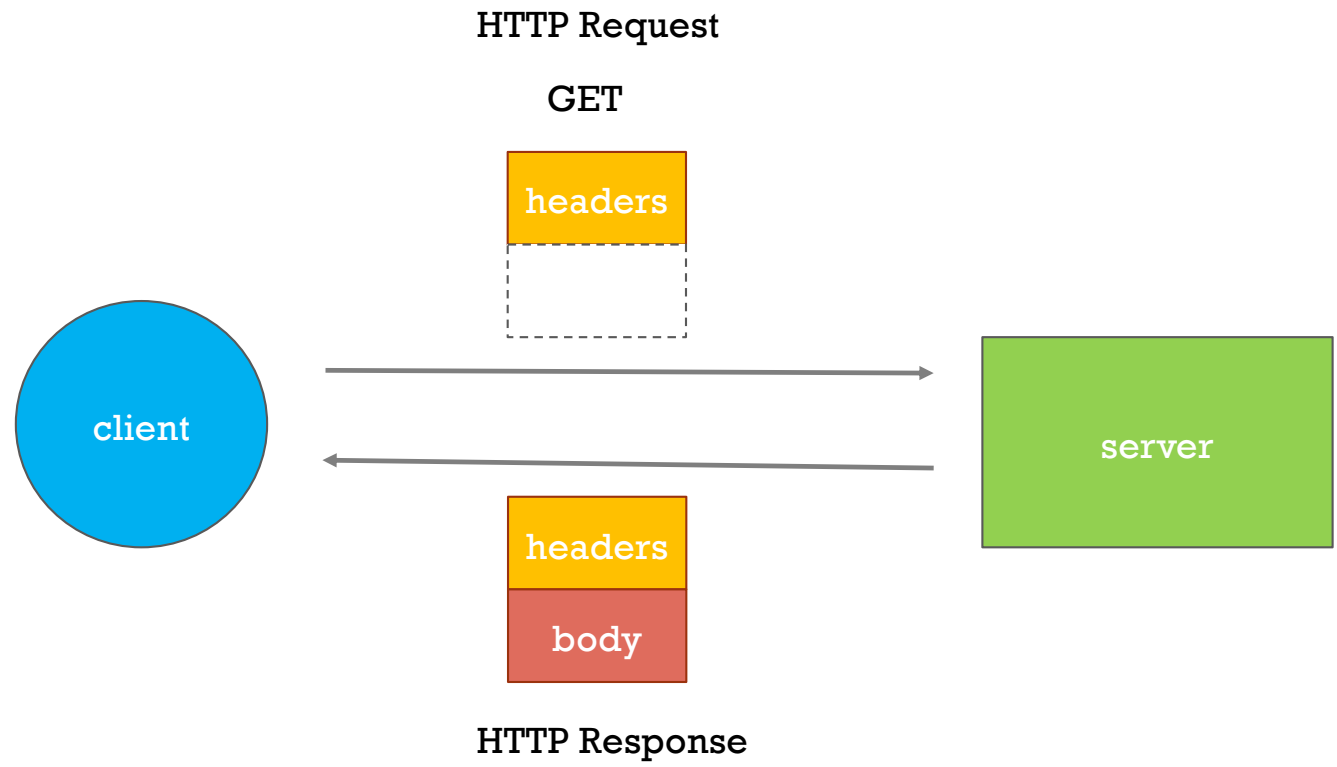
```
<pre>
  <?php
    print_r($_GET);
  ?>
</pre>
```

- 測試登入表單。
- \$\_REQUEST 的功能是？
- 如何用 Chrome 觀察表單的 get 和 post 方法？

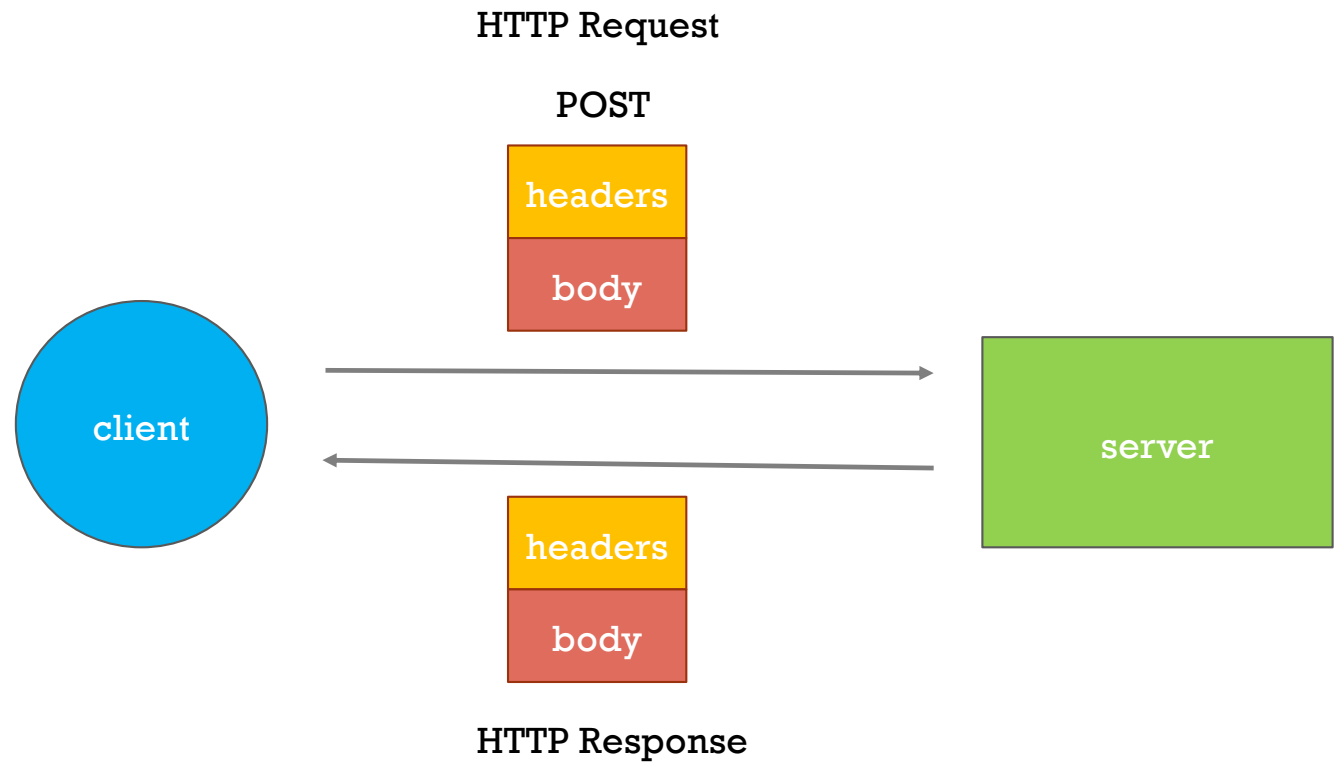




HTTP document



HTTP document





# 7. 函式

- 函式（function）就是有某種功能的小程式，可以看成是一個神奇的盒子，輸進一些資料後，可以得到處理過後的結果。
- 一個程式是為了解決某個大問題，要一下子解決一個大問題並不容易著手，把大問題分開成許多小問題，再個個擊破，就比較容易。
- 使用函式就是把問題分開個別解決。
- 之前使用過的 `phpinfo()`、`array()` 和 `count()` 都是 PHP 內建的函式。



# 7.1 自訂函式

- 自訂函式的語法以 function 關鍵字為開頭。

```
function 函式名(參數1, 參數2, ...) {  
    // 函式的內容敘述  
    return 回傳值;  
}
```

```
<?php  
function multi($a, $b=10) {  
    return $a * $b;  
}  
echo multi(6, 7). '<br />';    // 42  
echo multi(8). '<br />';    // 80
```



- 以參照為參數。

```
<?php
function swap(&$a, &$b) {
    $c = $a;
    $a = $b;
    $b = $c;
}
$m = 100;
$n = 'abc';
swap($m, $n);
echo "$m,  $n";
```



- 函式外的變數
- 在 PHP，函式內不能直接使用函式外設定的變數，若要使用必須在函式內宣告變數為全域變數，才能使用。
- 這樣的做法是，儘量避免在函式內使用外面的變數，可以增加函式的獨立性和再用性。

```
<?php
$g1 = 1000;
$g2 = 2000;

function fun() {
    global $g2;
    echo ">{$g1}< >{$g2}<";
}
fun();
```







## 8. 類別

- 類別可以看成是設計圖，依照設計圖實作出來的就是物件。類別也可以想成是使用者定義的資料類型，而物件是以類別宣告出來的變數。
- 簡單類別，若先不考慮繼承，類別的定義語法如下：

```
class 類別名稱 //類別定義的標頭
{
    // 屬性宣告
    // 建構函式定義
    // 方法定義
}
```



- 屬性就是物件的變數。
- 屬性的宣告可以用修飾字 var、public 和 private。var 和 public 的功能是相同的，都是定義可以公開使用的屬性；private 則是定義私有屬性，只能在類別裡使用。
- 下式用來說明 public 和 private 的區別。

```
<?php
class Person {
    var $name;
    public $mobile;
    private $sno = 'secret';
}
$p = new Person;
$p->name = "Shinder Lin";
$p->mobile = "0918123456";
// $p->sno = "123"; // 取用私有屬性將發生錯誤
print_r($p);
echo $p->name;
```



# 8.1 方法

- 「方法」其實就是函式，主要區別是：方法屬於物件，函式則是全域的。
- 在修飾字方面，使用 `private` 表示私有方法，預設為公開方法。
- 類別定義中，除了「宣告並指定屬性的敘述」可以直接放在類別內，其餘敘述都應該放在方法內。

```
class Person {  
    private $name;  
    function setName($n) {  
        $this->name = $n;  
    }  
    function getName() {  
        return $this->name;  
    }  
}  
$p = new Person;  
$p->setName("Victor");  
echo $p->getName();
```



## 8.2 建構函式及解構函式

- 建構函式是物件在使用 `new` 建立之後，會自動被呼叫的函式，用來做物件的初始設定之用。
- 建構函式名為「`__construct`」。
- 物件被刪除前會呼叫解構函式「`__destruct`」，有時資料庫的操作在物件刪除後就要關閉，即可在解構函式中進行關閉動作。
- `unset()` 刪除物件前，解構函式會被呼叫；程式結束前（刪除所有物件前）所有物件的解構函式都會被呼叫。



```
<?php
class Person {
    var $name;
    function __construct($n) {
        $this->name = $n;
        echo $this->name . '建立<br />';
    }

    function __destruct() {
        echo $this->name . '解構<br />';
    }
}
$p = new Person('p');
unset($p);
$q = new Person('q');
```



## 8.3 類別方法

- 有些方法沒有用到 `$this`，而且其目的是用來直接呼叫的，稱為類別方法。類別方法定義方式和一般方法一樣，不同處在於呼叫方式，使用「`::`」。

```
<?php
class MyMath {
    function distance($x1, $y1, $x2, $y2) {
        $dx = $x1 - $x2;
        $dy = $y1 - $y2;
        return sqrt($dx * $dx + $dy * $dy);
    }
}
echo MyMath::distance(1, 1, 11, 11);
```





# 9. Cookie

- Cookie 是網站透過瀏覽器將資料存放在硬碟裡的機制。基於安全性，Cookie 有幾個特性：
  1. 以網域為單位：例如 pchome.com.tw 不可讀取 yahoo.com.tw 所設定的 Cookie。
  2. 大小受限：不同的瀏覽器有不同的大小限制，一般為 4096 bytes。
  3. 客戶端可能可以看得到：IE 的 Cookie 是以檔案的形式存在。
  4. 客戶端可以設定：瀏覽器可以設定是否使用 Cookie，一般設定為使用，筆者也建議使用。
  5. 過期：Cookie 是有期限的，過期就會失效。
  6. 自動傳給主機：瀏覽器每次發 request 時，都會將有效的 Cookie 放在 HTTP Header 裡，傳給 server。
- Cookie 的內容應該要簡短，而且不應存放敏感性資料（例如：密碼、手機號碼）。
- Cookie 是好的，可以幫助網站做客製化的設定，大部份的 Session 機制也是建立在 Cookie 之上。





- PHP設定 Cookie 使用 setcookie() 函式：

```
setcookie("Cookie變數名稱", "Cookie數值", "期限", "路徑", "網域", "安全");
```

- 通常我們只會用到前三個參數：「Cookie名稱」、「Cookie值」和「期限」。
- 「期限」用的是 Unix 系統上的時間戳記（Unix timestamp），單位為秒，用 time() 函式可以取得當下的時間戳記。若不使用「期限」參數，瀏覽器的所有視窗關閉時為過期時間。
- PHP 讀取 Cookie 使用 \$\_COOKIE（預設的陣列變數）
- Cookie的設定（使用 setcookie() 函式）是放在HTTP的Header（表頭）裡，應該在HTML內容還沒有出現時設定。
- 若 PHP 的設定「output\_buffering」為「On」時，表示HTTP內容有做緩衝，或許不需要放在最前面。不過避免「output\_buffering」為「Off」，或者緩衝區已滿的情況下，setcookie() 函式應該在HTML內容還沒有出現時設定。。



```
<?php
if(! isset( $_COOKIE['mycookie'] ) ){
    setcookie("mycookie", "10", time()+30);
} else {
    setcookie("mycookie", $_COOKIE['mycookie'] + 1, time()+30);
}
?>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Cookie</title>
</head>
<body>
<?php
if(! isset( $_COOKIE['mycookie'] ) ){
    echo '第一次設定Cookie, 或重新設定';
} else {
    echo $_COOKIE['mycookie'];
}
?>
</body>
</html>
```





# 10. Session

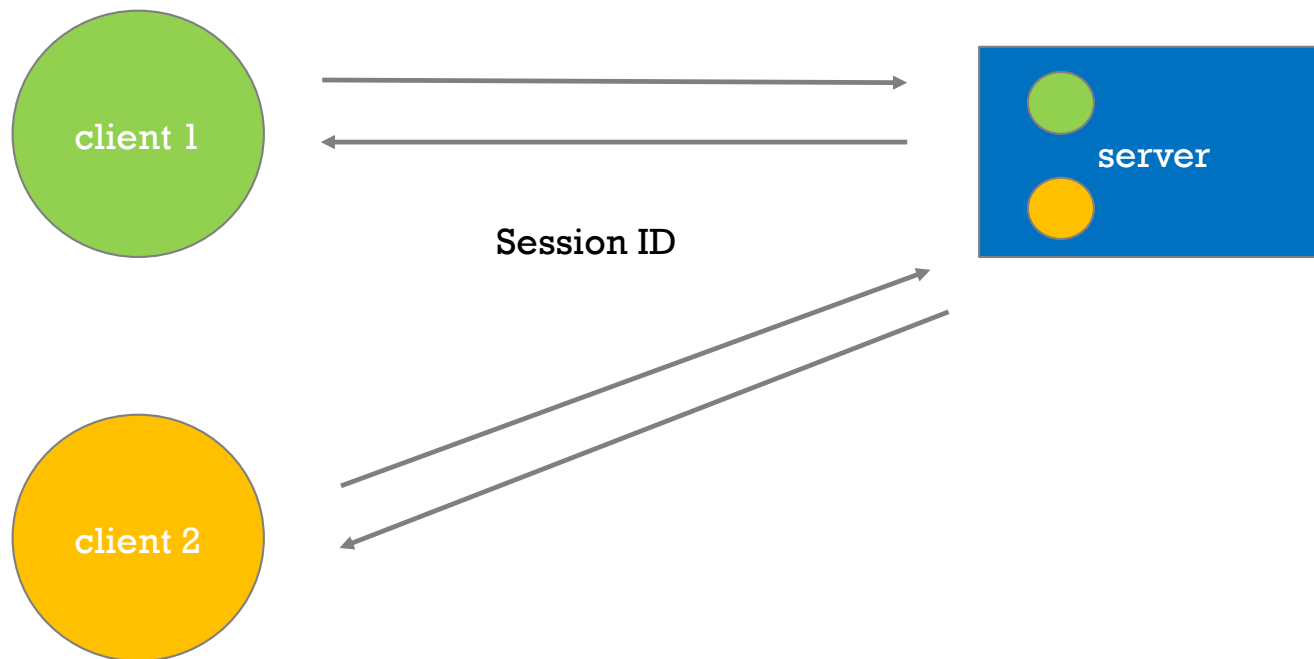
- 一般講「會談」（Session）是指客戶還在瀏覽該站，或者還在使用該站的服務。
- Session 是伺服器為了知道客戶端是否還在瀏覽該站的機制。
- 為了不佔用連線資源，HTTP 是個溝通完就斷線的協定，所以伺服器無法得知客戶端是否還在該站瀏覽。因此，就必須訂定一些規則來實現這個機制。
- 首先，server 必須先分辨 client，一般的做法是利用 Cookie 存放「Session ID」，只要在 client 第一次拜訪時將 Session ID 存入 Cookie 即可。此時 Session 的存活就會依賴 Cookie 是否有效而定。
- 若 client 的瀏覽器停在某個網頁，使用者可能某些原因（例如：去洗澡）久久未再拜訪該站，或者根本就已離開該站。此時會依 Session 的存活時間，決定 Session 是否有效。
- 當然，server 是以 client 最後一次拜訪開始計時的；若 client 在 Session 存活時間內，持續訪問該站，Session 就會一直有效。一般 Session 的存活時間會設定為 20 ~ 60 分鐘（有些網站會設定幾天）。



- 目前 PHP 預設存放 Session 資料的方式是「檔案」。
- 常用操作Session的函式有：
  - session\_start() 啟用 Session、
  - session\_destroy() 清除 Session 所有資料。
- 讀取和設定 Session 使用 \$\_SESSION 預設變數。session\_name 函式可以讀取或設定 Session ID 名稱，預設為 PHPSESSID。。

```
<?php
session_start(); // 啟用 session
if(! isset($_SESSION['num'])){
    $_SESSION['num'] = 1;
} else {
    $_SESSION['num'] ++;
}
// unset($_SESSION['num']); // 刪掉某個 session 變數
echo $_SESSION['num'];
```







# 11. 資料庫連線 ( PDO )

```
<?php
$db_host = 'localhost';
$db_name = 'proj57';
$db_user = 'root';
$db_pass = 'admin';

$dsn = "mysql:host={$db_host};dbname={$db_name};charset=utf8"; // data source name

$pdo_options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC
];
try {
    $pdo = new PDO($dsn, $db_user, $db_pass, $pdo_options);
} catch(PDOException $ex) {
    echo 'Connection failed: '. $ex->getMessage();
}
```





## 11.1 使用PDO::query

```
<?php
require __DIR__ . '/__connect_db.php';

// 取得分類資料
$c_sql = "SELECT * FROM categories WHERE parent_sid=0";
$c_stmt = $pdo->query($c_sql);
$cates = $c_stmt->fetchAll(PDO::FETCH_ASSOC);
```



## 11.2 使用PDO::prepare

```
// 去掉頭尾空白，然後轉小寫
$email = strtolower(trim($_POST['email']));
// 密碼編碼，不要明碼
$password = sha1(trim($_POST['password']));

$sql = "SELECT `id`, `email`, `mobile`, `address`, `birthday`, `nickname`
        FROM `members` WHERE `email`=? AND `password`=?";
$stmt = $pdo->prepare($sql);
$stmt->execute([
    $email,
    $password,
]);
$user = $stmt->fetch(PDO::FETCH_ASSOC);
```





# 12. 通訊錄管理

## 12.1 計算總筆數

```
$t_sql = "SELECT COUNT(1) FROM address_book";  
$total = $pdo->query($t_sql)->fetch(PDO::FETCH_NUM)[0];
```



## 12.2 依頁碼讀取資料

```
$per_page = 5; // 每頁幾筆
$page = isset($_GET['page']) ? intval($_GET['page']) : 1; // 用戶要看第幾頁

$total_pages = ceil($total/$per_page); // 總頁數
$page = $page > $total_pages ? $total_pages : $page;
$page = $page < 1 ? 1 : $page;

$sql = sprintf("SELECT
    `sid`, `name`, `email`, `mobile`, `address`, `birthday`
    FROM address_book
    ORDER BY `sid` DESC
    LIMIT %s, %s", ($page-1)*$per_page, $per_page);
$stmt = $pdo->query($sql);
$ar = $stmt->fetchAll(PDO::FETCH_ASSOC); // 一次讀取全部
```



## 12.3 分頁按鈕 pagination

```
<ul class="pagination">
  <?php for ($i = 1; $i <= $total_pages; $i++): ?>
    <li class="page-item <?= $i == $page ? 'active' : '' ?>">
      <a class="page-link" href="?page=<?= $i ?>"><?= $i ?></a>
    </li>
  <?php endfor ?>
</ul>
```



## 12.4 分頁按鈕參考 ( JavaScript )

```
(function (page, totalPages, prevNum) {  
    let beginPage, endPage;  
    if (totalPages <= prevNum * 2 + 1) {  
        beginPage = 1;  
        endPage = totalPages;  
    } else if (page - 1 < prevNum) {  
        beginPage = 1;  
        endPage = prevNum * 2 + 1;  
    } else if (totalPages - page < prevNum) {  
        beginPage = totalPages - (prevNum * 2 + 1);  
        endPage = totalPages;  
    } else {  
        beginPage = page - prevNum;  
        endPage = page + prevNum;  
    }  
    output.beginPage = beginPage;  
    output.endPage = endPage;  
})(page, output.totalPages, 3);
```



## 12.5 在選單上標示目前頁面

```
<ul class="nav navbar-nav">
  <li class="<?= $page_name == 'data_list' ? 'active' : '' ?>">
    <a href="data_list.php">資料列表</a>
  </li>
  <li class="<?= $page_name == 'data_insert' ? 'active' : '' ?>">
    <a href="data_insert.php">新增資料</a>
  </li>
</ul>
```





## 12.6 新增資料

```
$sql = "INSERT INTO `address_book`(  
    `name`, `email`, `mobile`, `address`, `birthday`  
    ) VALUES (?, ?, ?, ?, ?)";  
$stmt = $pdo->prepare($sql);  
$stmt->execute([  
    $_POST['name'],  
    $_POST['email'],  
    $_POST['mobile'],  
    $_POST['address'],  
    $_POST['birthday']  
]);  
  
if($stmt->rowCount() == 1){  
    // 資料新增完成  
}
```

\*\* 使用 `prepare()` 和佔位符號 `?` 可以自動跳脫單引號排除 SQL injection



## 12.7 JS檢查 email 格式

```
function checkForm() {
    if (document.form1.name.value.length < 2) {
        alert('請填寫正確姓名');
        return false;
    }
    if (!validateEmail(document.form1.email.value)) {
        alert('email 格式不正確');
        return false;
    }
}
// http://stackoverflow.com/questions/46155/validate-email-address-in-javascript
function validateEmail(email) {
    var re =
/^((([^<>()\\[\]\\. ,;: \s@"]+(\.[^<>()\\[\]\\. ,;: \s@"]+)*)|("[.+"])|(\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\]|([a-zA-Z\d-0-9]+\.)+[a-zA-Z]{2,}))$/;
    return re.test(email);
}
```



## 12.8 PHP 驗證 email 格式

```
<?php
# https://www.php.net/manual/en/filter.examples.validation.php

$email_a = 'joe@example.com';
$email_b = 'bogus';

if (filter_var($email_a, FILTER_VALIDATE_EMAIL)) {
    echo "Email address '$email_a' is considered valid.\n";
}
if (filter_var($email_b, FILTER_VALIDATE_EMAIL)) {
    echo "Email address '$email_b' is considered valid.\n";
} else {
    echo "Email address '$email_b' is considered invalid.\n";
}
```



## 12.9 移除資料

```
<?php
require __DIR__ . '/__connect_db.php';
$cid = isset($_GET['cid']) ? intval($_GET['cid']) : 0;
if (empty($cid)) {
    header('Location: ./data_list.php');
    exit;
}
$sql = "DELETE FROM `address_book` WHERE `cid`=$cid";
$stmt->execute($sql);

if (isset($_SERVER['HTTP_REFERER'])) {
    // 從哪裡來回哪裡去
    header('Location: ' . $_SERVER['HTTP_REFERER']);
} else {
    header('Location: ./data_list.php');
}
```



## 12.10 修改資料

```
$sql = "UPDATE `address_book` SET
        `name`=?, `email`=?, `mobile`=?, `address`=?, `birthday`=?
        WHERE `sid`=?"
$stmt = $pdo->prepare($sql);
$stmt->execute([
    $_POST['name'],
    $_POST['email'],
    $_POST['mobile'],
    $_POST['address'],
    $_POST['birthday'],
    $_POST['sid']
]);

// 影響的列數 (筆數)
if ($stmt->rowCount() == 1) {
    echo '資料修改完成';
}
```



## 12.11 避免 XSS 攻擊

```
<td><?= strip_tags($row[ 'address' ]) ?></td>
```

```
<td><?= htmlentities($row[ 'address' ]) ?></td>
```



## 12.12 PHP 產出 JSON

```
<?php
$ar = [
    'name' => '彼德潘 / peter',
    'age' => 30,
    'others' => array(56, 'aaa', 72),
];

echo json_encode($ar, JSON_UNESCAPED_UNICODE);
```



## 12.13 兩層分類選單參考

```
$stmt = $pdo->query("SELECT * FROM categories ORDER BY sid DESC");
$rows = $stmt->fetchAll();
$first = [];

foreach($rows as &$r){
    if($r['parent_sid']==0){
        $first[] = &$r; #把第一層的資料放到陣列裡
    }
}

foreach($first as &$f){
    foreach($rows as &$r){
        if($f['sid']==$r['parent_sid']){
            $f['children'][] = &$r; #把第二層的資料放到陣列裡
        }
    }
}

echo json_encode($first);
```





## 12.14 多層選單結構組成

```
function getCateTree($pdo){
    $sql = "SELECT * FROM `categories` ORDER BY `parent_sid`, `sid`";
    $rows = $pdo->query($sql)->fetchAll();
    $dict = [];
    foreach ($rows as &$v) {
        $dict[$v['sid']] = &$v;
    }
    $cateTree = [];
    foreach ($dict as &$row) {
        if ($row['parent_sid'] != 0) {
            $dict[$row['parent_sid']]['children'][] = &$row;
        } else {
            $cateTree[] = &$row;
        }
    }
    return $cateTree;
}
echo json_encode( getCateTree($pdo) );
```



## 12.15 用戶密碼編碼

- ◆ password\_hash()

<https://www.php.net/manual/en/function.password-hash>

- ◆ password\_verify()

<https://www.php.net/manual/en/function.password-verify>



**THANK YOU**

