PHP網站程式設計

授課講師 董淑蕙

教材編寫 董淑蕙

課程綱要

- 單元一：WWW Application簡介
- 單元二：PHP伺服端網頁程式設計簡介
- 單元三：PHP的安裝與設定
- 單元四：PHP的程式架構與基本語法
- 單元五：PHP的資料型別與變數
- 單元六：PHP的運算式
- 單元七：PHP的流程控制

課程綱要

- 單元八：PHP的陣列介紹
- 單元九：PHP的函數介紹
- 單元十：PHP存取MySQL資料庫
- 單元11：網頁之間的資訊傳遞
- 單元12：上傳檔案

課程所需知識

- Windows作業系統
- IIS之基本管理
- MySQL資料庫之管理
- HTML網頁設計
- 程式語言能力

- Windows 作業系統
- Wampserver
  - Apache X.X.X
  - MySQL 8.X.X
  - PHP X.X.X

單元一： WWW Application簡介

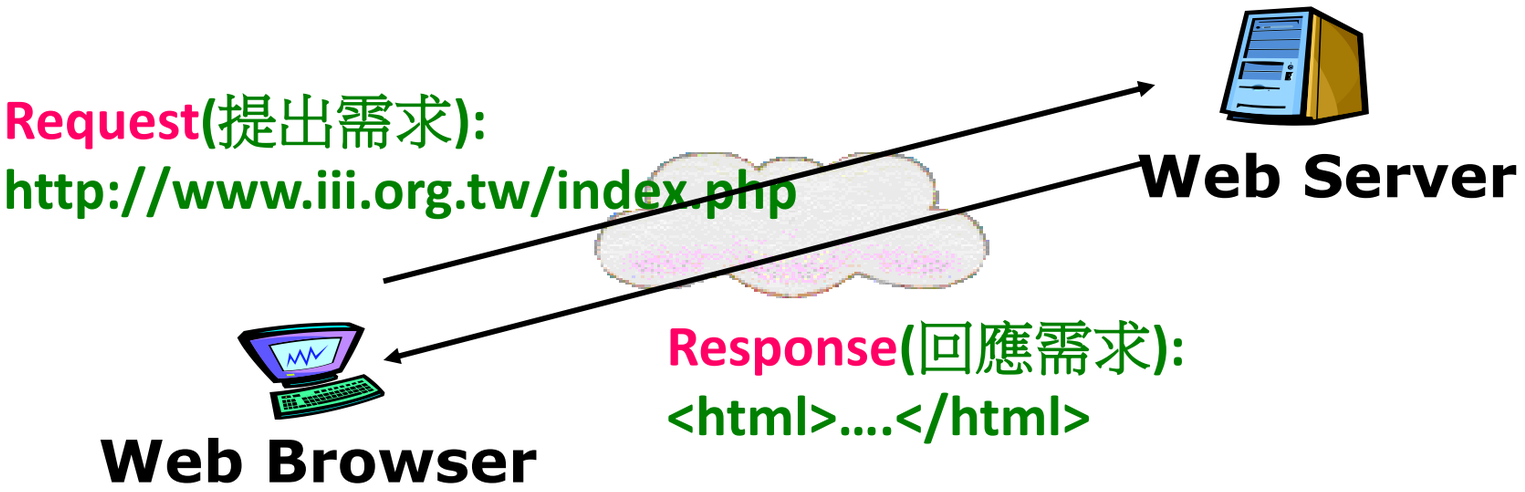
- WWW Application架構
- Web application客戶端(client)的網頁技術
- Web application伺服端(Server)的動態網頁技術

WWW Application架構_1

- World Wide Web全球資訊網
- 伺服器(Server)上置放許多的資訊供客戶端(Client)來擷取
- 客戶端使用HTTP通訊協定(HyperText Transfer Protocal)提出請求
- 伺服端使用HTTP通訊協定來回覆客戶端所要的網頁和相關的檔案

WWW Application架構_2

- 伺服端提供Web服務
  - IIS
  - Apache
- 客戶端提出服務請求
  - 瀏覽器



Web application客戶端(client)的網頁技術

- 主要有
  - HTML：描述網頁的內容與結構
  - CSS：美化網頁的外觀
  - JavaScript：可在瀏覽器中展現動態而豐富的網頁，並讓使用者能與網頁有更好的互動性
- 衍生的相關技術有
  - Bootstrap工具
  - jQuery函式庫
  - 框架
  - More...

Web application伺服端(Server)的動態網頁技術

- 客戶端動態網頁程式設計雖可做各項動態的圖文整合與展現，但却無法讓使用者和資料庫中的資料去做互動，如線上查詢、網路購物、群組討論及訪客留言等等，透過伺服端的動態網頁程式設計即可完成這些工作。伺服端常使用的語言有
  - PHP
  - Asp.net
  - Jsp，Java Servlet
  - Python
  - Node.js
  - More…

單元二：PHP伺服端網頁程式設計簡介

- PHP的歷史
- PHP的特色
- PHP的運作環境

PHP的歷史_1

- PHP首版 (1995)

Rasmus Lerdorf

1995年Rasmus Lerdorf用Perl開發的一套工具，稱為Personal Home Page Tools。用來維護自己的網頁及追踪並統計訪客的使用狀況。

PHP的歷史_2

- PHP/FI 1.0 (1996)
  - 之後Rasmus Lerdorf改用C語言撰寫此套工具，並加入連結資料庫及建立動態網站的功能，稱為PHP/FI(Personal Home Page/Forms Interpreter)
- PHP/FI 2.0 (1997)
  - Rasmus Lerdorf釋出PHP/FI 1.0，提供社群來共同加速開發，於1997年推出PHP/FI2.0版，有更多的變數與表單處理功能，成為PHP的部分雛形。

PHP的歷史_3

- PHP 3.0 (1998)
  - PHP FI2.0之後， Rasmus Lerdorf將其轉移給 Andi Gutmans和 Zeev Suraski，被完全改寫成官方版的PHP3.0(1998年)，提供了更多的擴展性，稱為Hypertext Preprocessor

Andi Gutmans

Zeev Suraski

PHP的歷史_4

- PHP 4.0 (2000)
  - Andi Gutmas和 Zeev Suraski後來又改寫 PHP以 Zend Engine為核心，成為PHP4.0(2000年)，支援更多的平台和 Web伺服器、HTTP的 Session管理和輸出緩衝區的功能等
- PHP 5.0 (2004)
  - 2004年推出新版的PHP5.0功能，支援物件導向程式設計，並提供更多的擴充函數
- PHP 6.?
  - 2005年時，PHP社群提出創建 PHP6以解決 Unicode的支援問題，但在開發社群中對多國語言支援的爭論和效能議題一直有著衝突與爭議，計畫正式在2010年宣告失敗而終止。

PHP的歷史_5

- PHP 7.0 (2015)
  - Facebook為了提昇自家網站的速度而重新改寫了 PHP編譯引擎。此舉引起了PHP開發者戮力思考 PHP的效能問題，並決定重新建構PHP。
  - 2015年12月推出PHP 7.0。大幅減少了記憶體使用、一致性的64位元支援及例外處理架構優化等等，在效能上至少加速了一倍。

PHP的特色

- 是一種跨平台的腳本語言(Script Language)，同時支援Linux/UNIX、Windows和Mac OS等作業系統
- 支援多種網站伺服器：Apache、IIS...
- 支援多種資料庫系統：MySQL、MS-SQL、ORACLE、IBM DB2、Sybase、Informix、ODBC...
- 支援多種通訊協定:HTTP、LDAP、IMAP...
- 支援session及cookie的機制
- 提供許多的函式庫
- 屬於開放原始碼(Open Source)
- 可以內嵌在HTML文件中，也可以獨立存檔

單元三：Wampse的安裝與設定

- 下載Wampserver
- 確定目前的windows系統中port number 80和3306没有其它服務佔用
- 安裝Wampserver
- Wampserver管理頁面
- 建立php新專案
- 寫一支hello.php, 測試是否可以正常執行

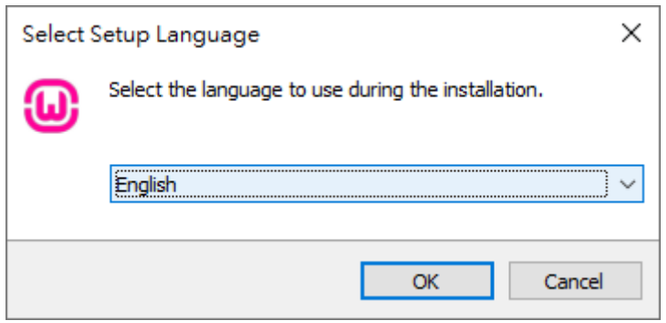
確認port number 80和3306没有被服務佔用

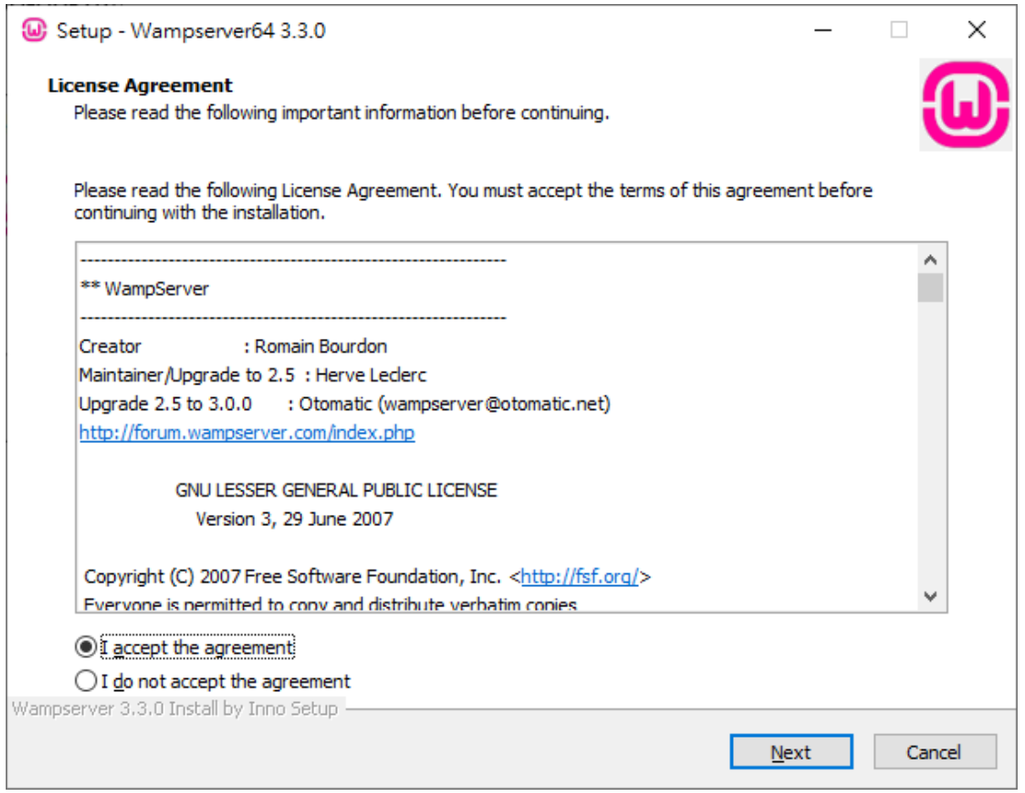
- 檢查埠號(port number) 是否已被佔用
  - Cmd命令列下指令netstat –a –b
- 若已先裝好IIS在port 80上
  - 更改IIS埠號
  - 或停止IIS服務
  - 或移除IIS服務
- 若已先裝好mysql
  - 更改已安裝好的mysql之port號
  - 或停止mysql服務
    - 請先將mysql**服務狀態**設定為**停止**
    - 並設定為**已停用**
  - 或移除已安裝好的mysql

安裝Wampserver_1

- Wampserver的官方網站：
https://www.wampserver.com/en/download-wampserver-64bits/

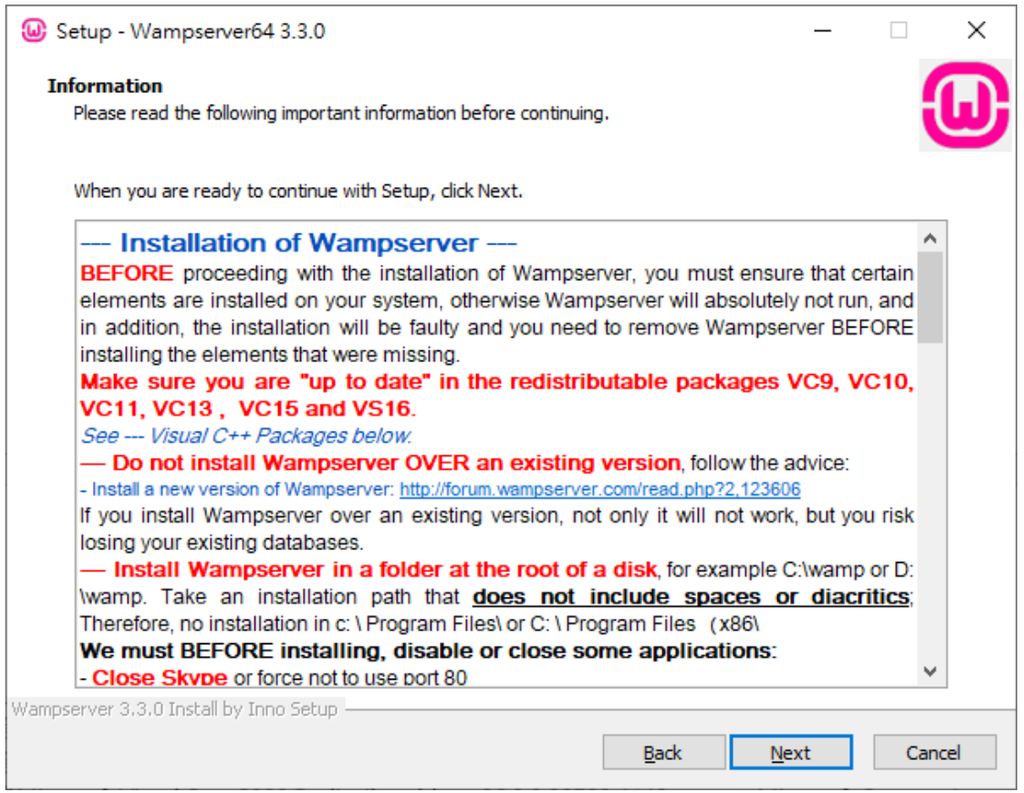
- 下載後執行

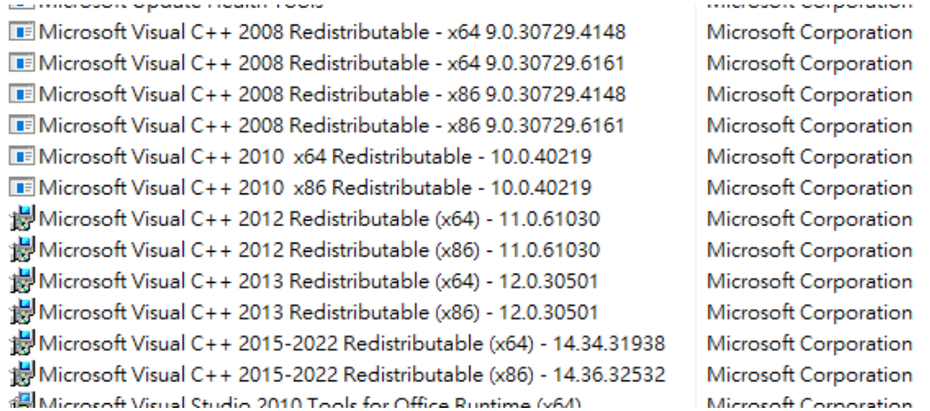




安裝Wampserver_2

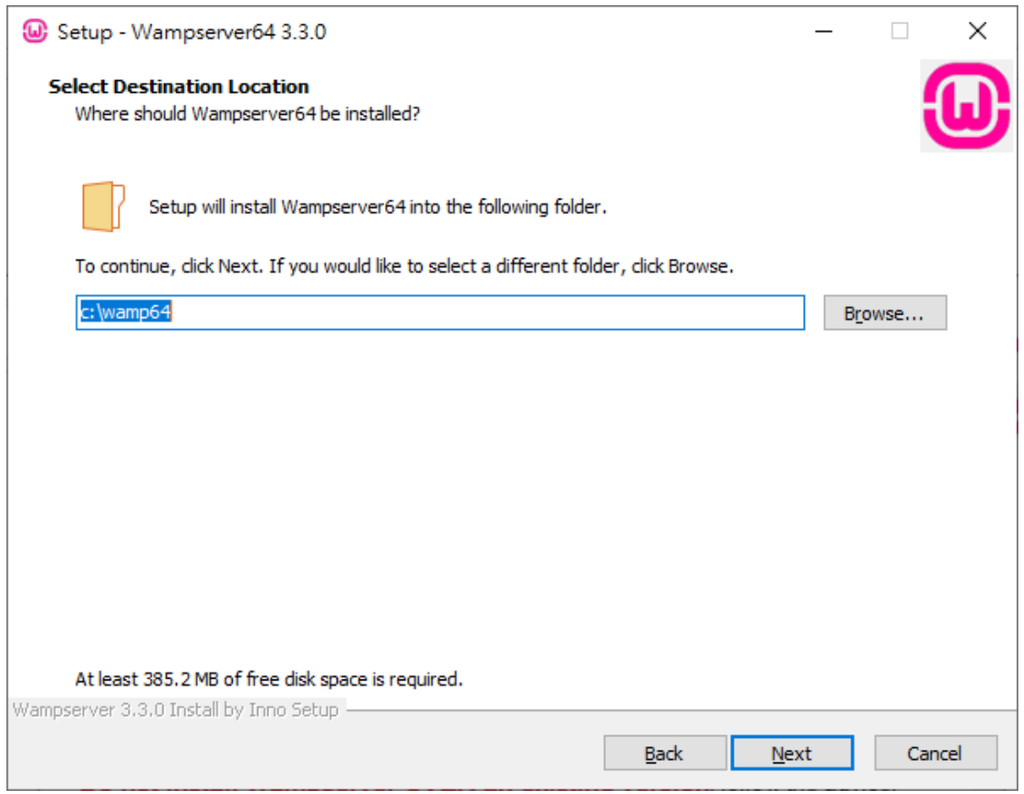
- 會提示作業系統中必需已安裝的項目...,視窗內的捲軸往下捲可看到必須升級到相關的VC++版本





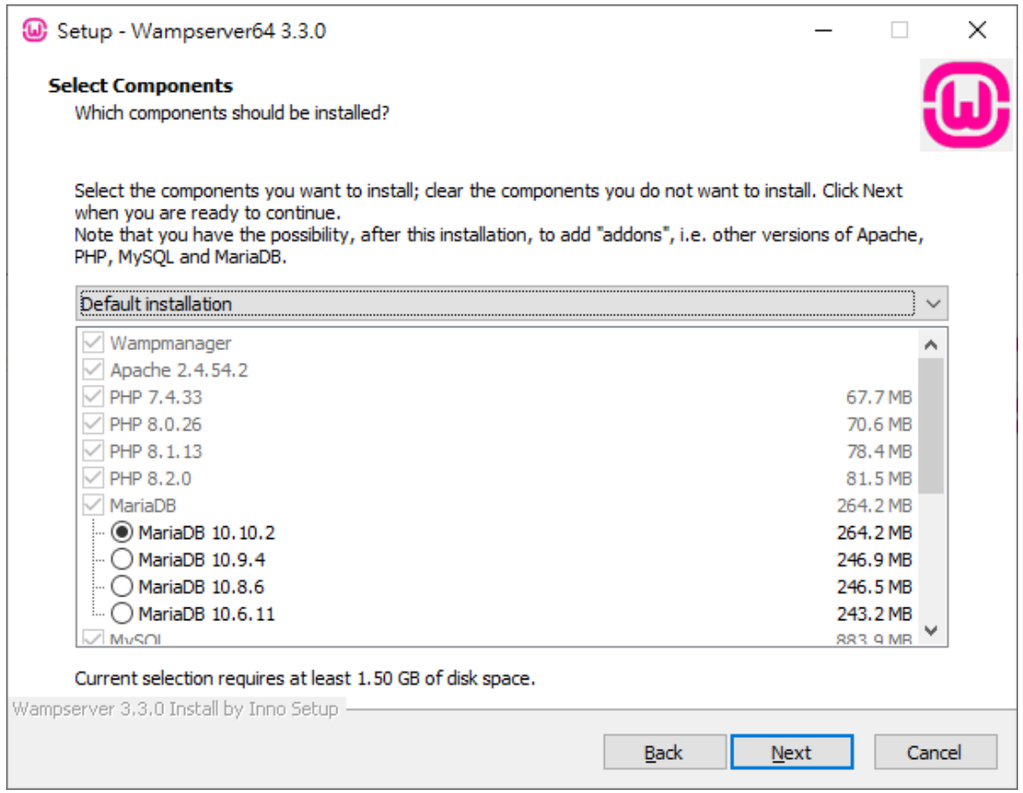
安裝Wampserver_3

- 詢問要安裝的路徑, 請使用預設路徑



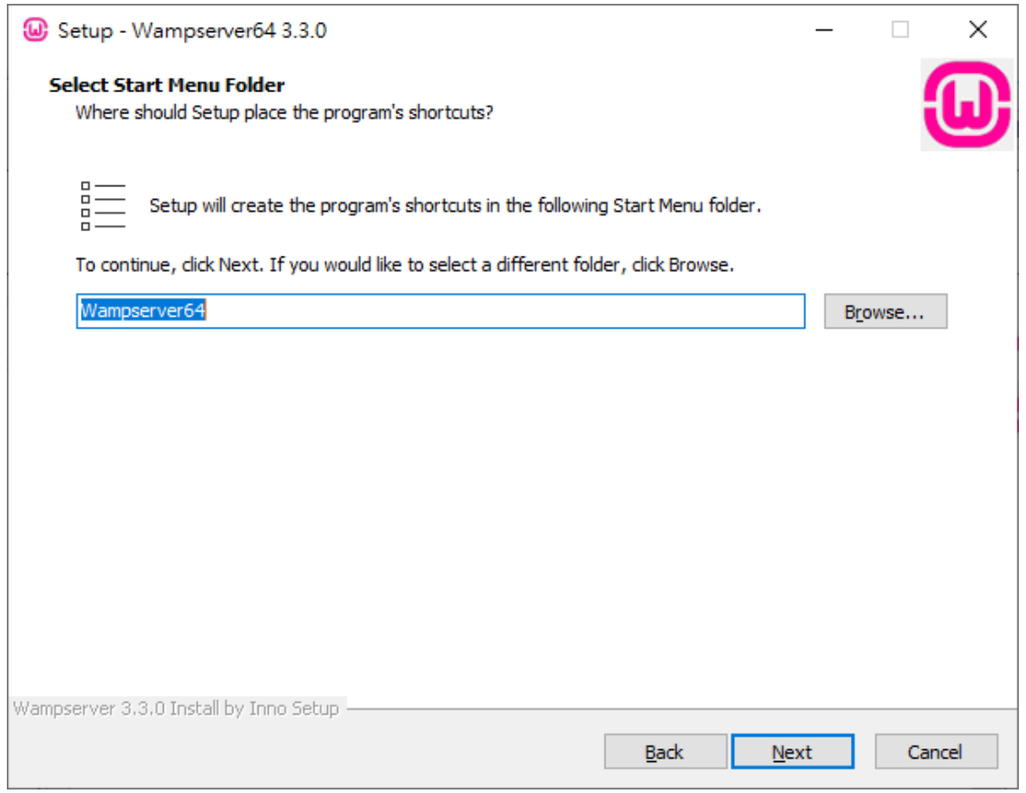
安裝Wampserver_4

- 詢問要安裝的項目, 請使用預設要安裝的項目



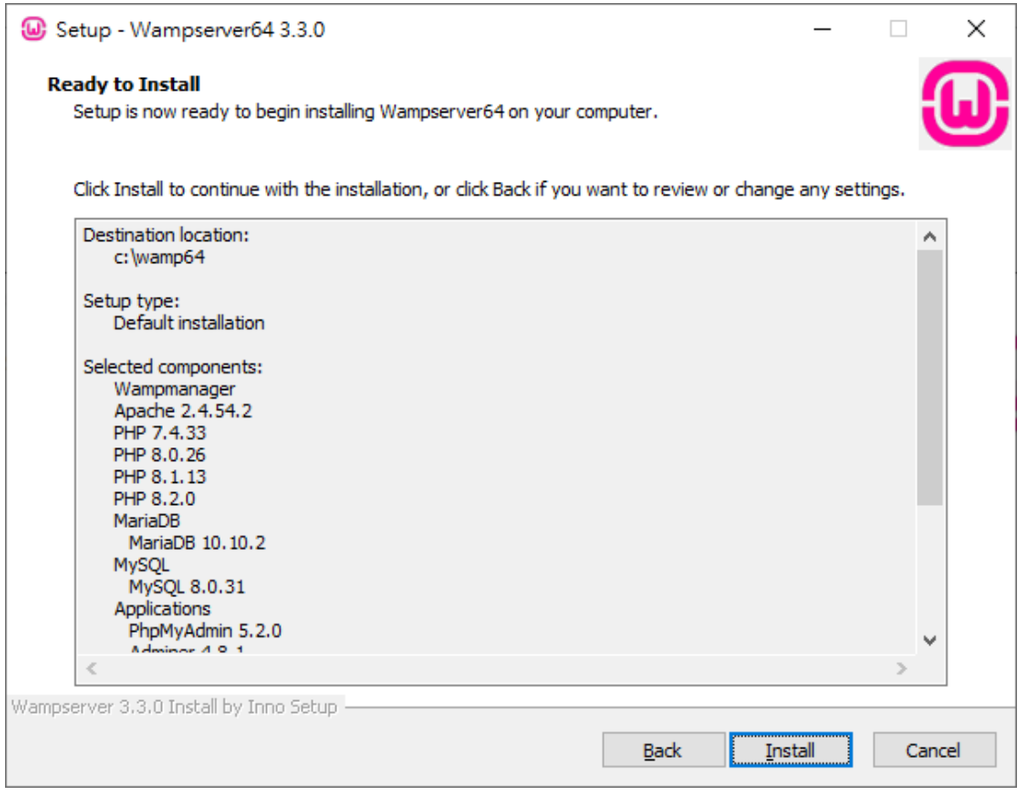
安裝Wampserver_5

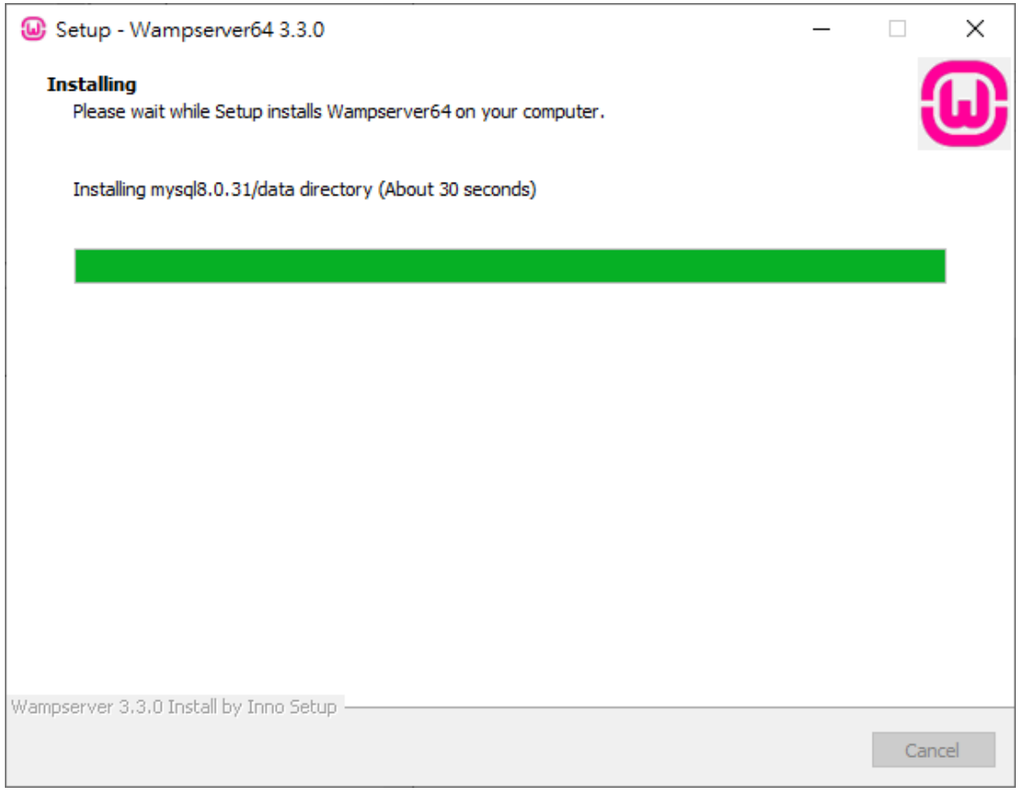
- 詢問在start menu中的資料夾名稱, 請使用預設名稱。



安裝Wampserver_6

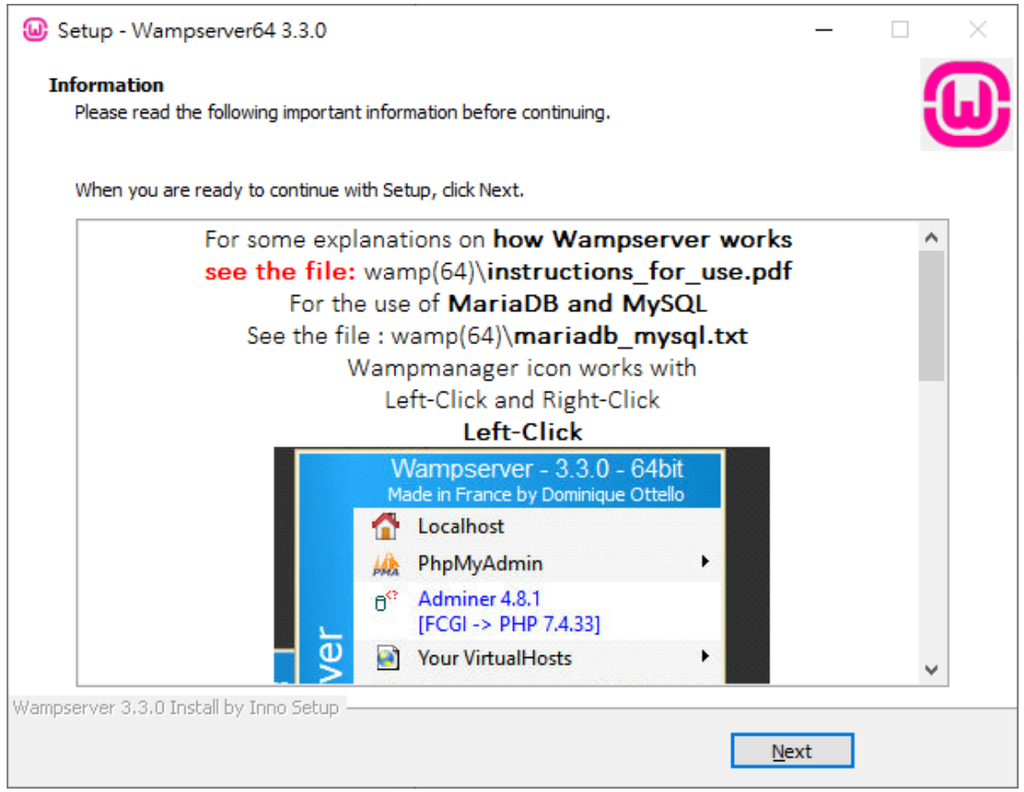
- 點按install開始安裝。
- 請注意，稍後會詢問所要使用的browser 是否為Edge

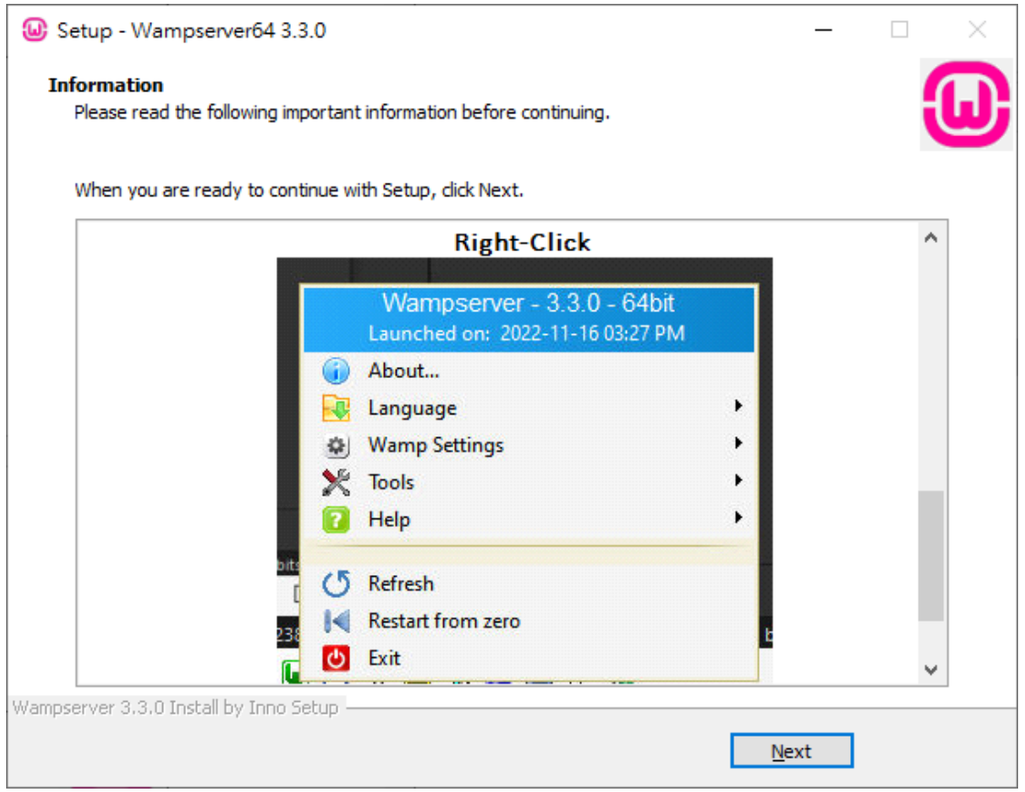




安裝Wampserver_7

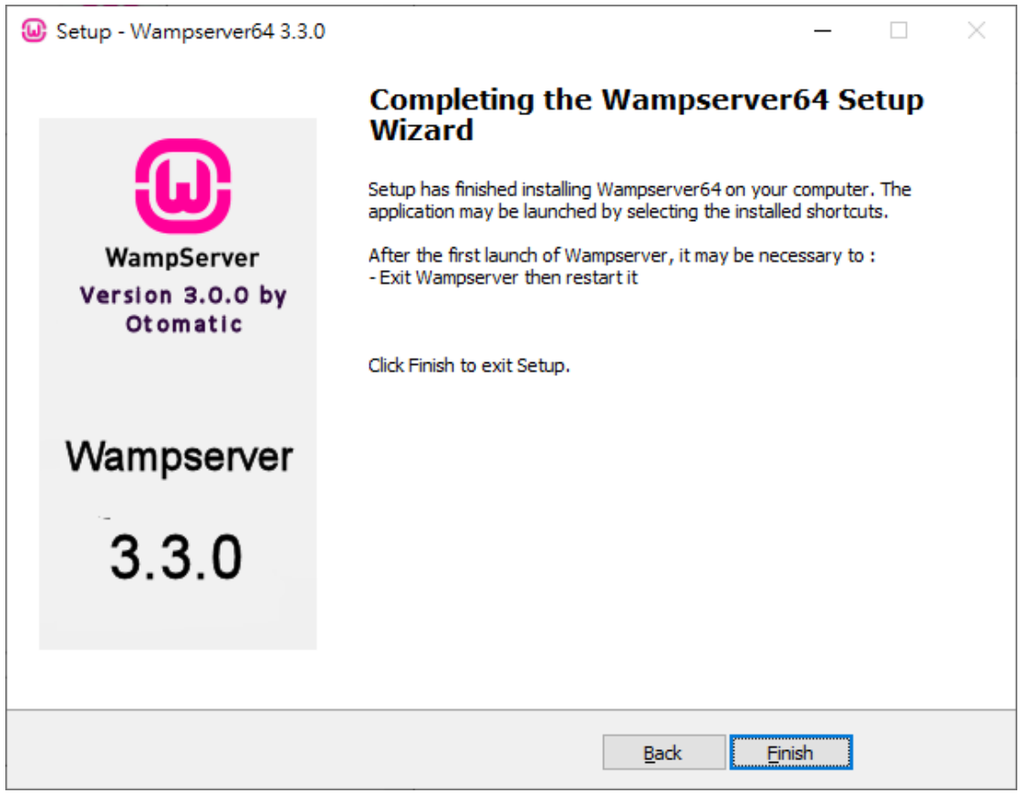
- 安裝完會跳訊息窗,說明按左鍵可以做什麼,按右鍵可以做什麼





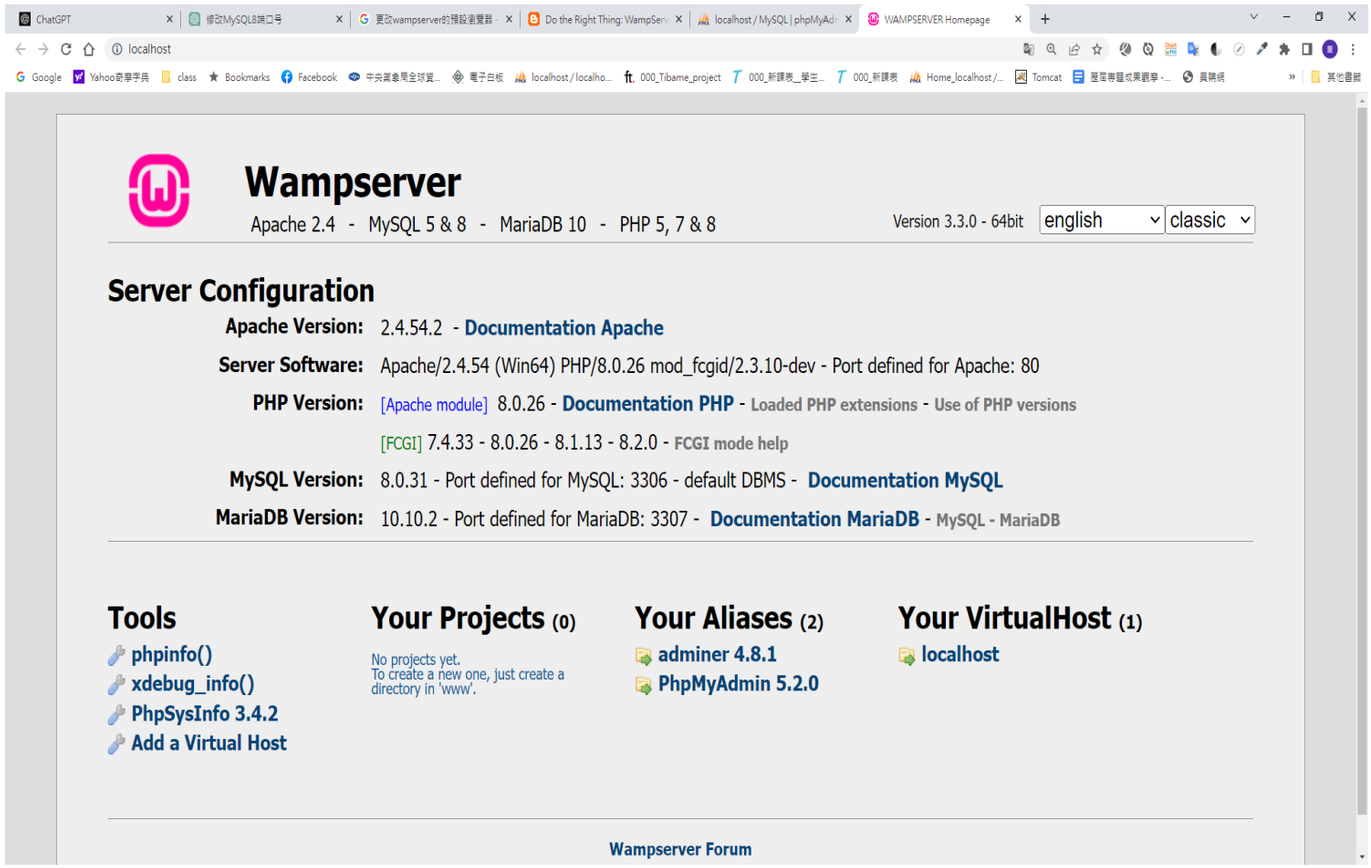
安裝Wampserver_8

- 安裝完畢



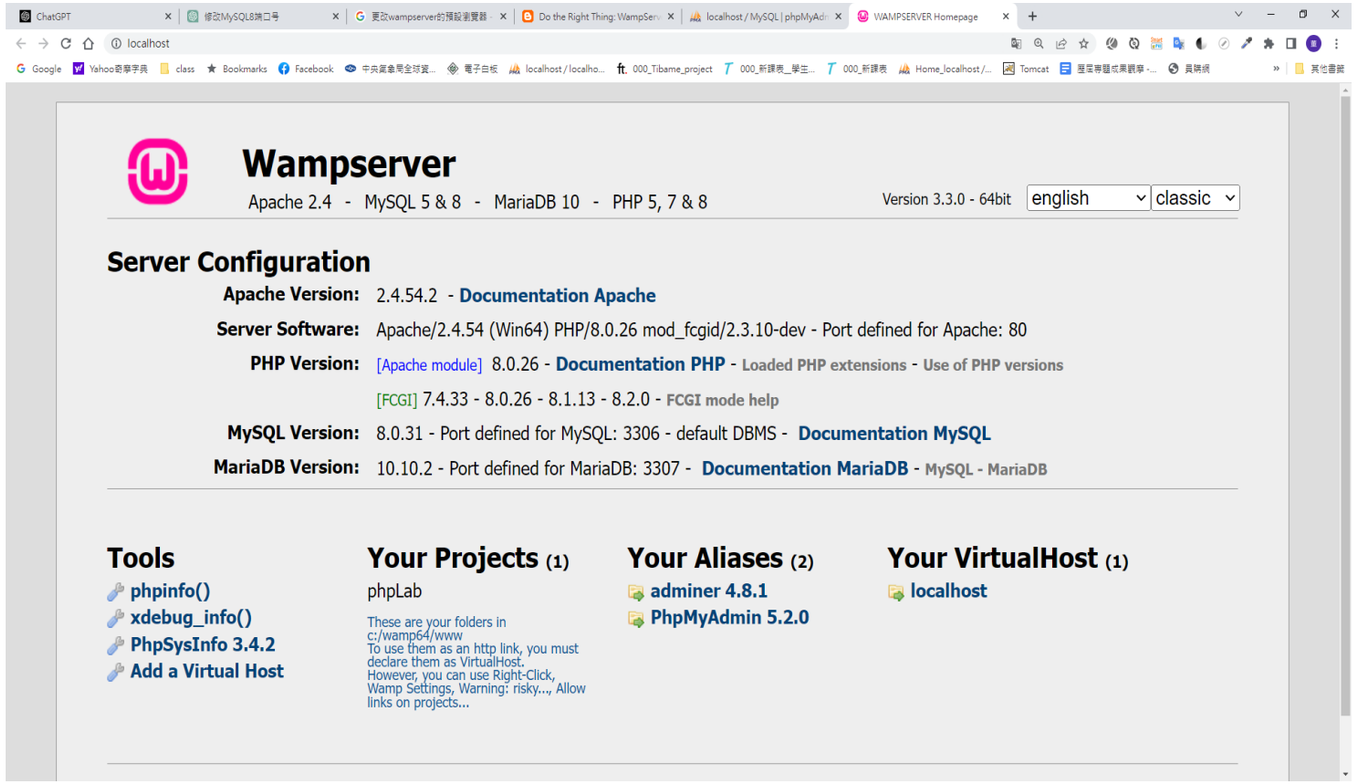
Wampserver管理頁面

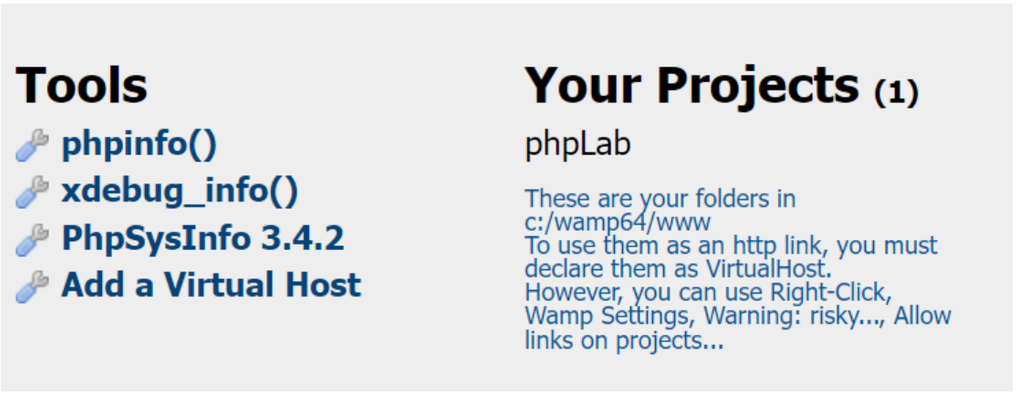
- 啟動wamp server ,圖示上按左鍵點選localhost 管理頁面的首頁



建立php新專案

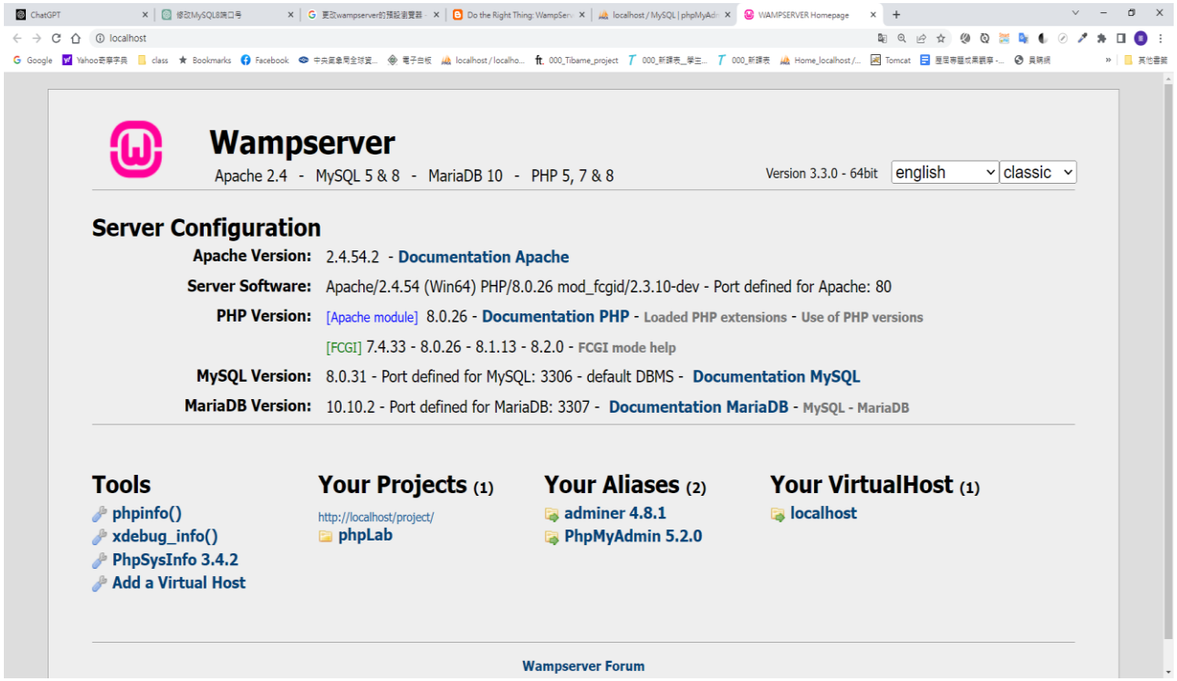
- 檔案總管中新增phpLab資料夾, 刷新管理介面, 可看見新的專案。但無法連結進入。





設定可以連結進專案

- 在Wampserver圖示上
  - 按右鍵
  - 點選Wamp settings
  - 點按Caution :risky! Only for experts.
- 再刷新管理頁面，如下，即可連結進專案



寫一支hello.php, 測試是否可以正常執行

- 在phpLab目錄下
  - 建立一個測試檔hello.php

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

  - 以瀏覽器執行該程式(localhost/phpLab/hello.php)

單元四：PHP程式架構、語法與網頁輸出

- PHP的程式架構
- PHP的語法
- PHP的標籤格式
- 輸出網頁內容

PHP的程式架構

```
<body>
 <?php
  //單行註解
  /* …
  多行註解
  … */
  變數宣告;
  函式宣告;
  敘述;
 ?>
</body>
```

- 附檔名為php
- 可內嵌在HTML結構中
- 也可另外存檔
  （副檔名可為.php或.inc檔）

PHP的程式範例

- 程式範例

```
<html>
 <head>
 <meta charset= “utf-8 “>
 <title>php程式架構</title>
 </head>
 <body>
 <?php
   echo "Hello world...<br>";
 ?>
   </body>
</html>
```

PHP語法

- PHP敘述(或稱指令)以"; "來代表敘述結束
- 單行註解

```
//註解
```

- 多行註解

```
/*
  註解...
*/
```

- 註解不會被執行

PHP的標籤格式

- 格式一　(建議使用)

```
<?php
echo "Hello world";
?>
```

- 格式二

```
<?
echo "Hello world";
?>
```

輸出網頁內容：使用echo敘述

- echo "資料1"[,"資料2"[,...]];
- 可以在輸出資料時整合HTML或CSS
- 例：

```
<?php
    echo "Hello world";
    echo "Welcome to PHP..";
    echo "<br>";
    echo "Welcome to HTML<br>";
    echo "Welcome to JavaScript...";
?>
```

單元五：PHP的資料型別與變數

- 變數
- 資料型別
- 變數的有效範圍
- 外來、內建與環境變數
- 表單欄位變數
- 常數
- 資料型別的檢查
- 型別轉換

變數

- 宣告一個儲存空間，用來存放會變動的資料
- 需以$開頭
- 語法：$變數名稱=值;
- 變數名稱大小寫視為不同
- 屬於弱型別，不需事先宣告，指定值給變數時即決定變數的資料型別
  - 如：$name= "Sara";

輸出網頁內容：輸出變數值之簡化版

- PHP提供輸出變數之簡化版，將變數夾在<?=　?>之間

```
<?php
  $name = "Sara";
   echo "姓名：", $name, "<br>";  //敘述A
?>
姓名：<?=$name?><br>  <!-- 同敘述A -->
```

PHP組態檔(php.ini)

- php.ini：PHP執行環境組態檔
- PHP允許開發者在此檔中設定自己所要的執行環境。
- PHP啟用時會尋找是否有php.ini檔
  - 若有找到此檔，則讀取它並執行該檔中的設定
  - 若讀不到此檔，PHP採用其預設值
- 若在PHP啟用後再修改此檔中的設定，則必須重啟Web伺服器才能讀到最新的設定。

PHP組態檔(php.ini)_3

- PHP短標籤寫法

```
<?
echo "Hello world";
?>
```

  - 若有php.ini檔，則其中的short_open_tag必需設定為On時，才可以使用短標籤寫法

  - 若没有php.ini檔，則預設是可以使用PHP短標籤寫法

資料型別簡介

- 定義出資料儲存所需要的空間及將來可被操作的方法
- 型別種類:
  - 強型別(strong type):資料要使用之前必須先宣告
  - 弱型別(weak type):資料要使用之前無需事先宣告,php為弱型別

資料的literal(字面)表示法

- 整數：321、0321、0x321
- 浮點數：123.456
- 字串: "PHP Bible" 、'讚!!!'
- 布林: true、false

資料型別 - 整數 : integer

- 整數：
  - 十進位：如123、-32
  - 八進位：以「0」或「0o」開頭的整數值，每一個位數的值為0~7之間。
    如025 (八進位) =>21 (十進位) 。
  - 十六進位：以「0x」開頭的整數值，位數值為0~9和A~F。
    如0x25 (16進位) =>37 (十進位)
  - 二進位：以「0b」開頭的整數值，每一個位數的值為0~1之間。
    如0b1010 (二進位) =>10 (十進位) 。
  - 使用底線(_)提高字面值的可讀性: PHP 7.4.0 開始，整數文字可以在數字之間包含底線 (_)
- 練習：

```
echo 025 , "<br>";    //八進位整數，會顯示十進位整數21
echo 0o25 , “<br>” ;  //八進位整數，會顯示十進位整數21, (自PHP 8.1.0開始)
echo 0x25 , "<br>";   //十六進位整數，會顯示十進位整數37
echo 0b1010 , "<br>";  //二進位整數，會顯示十進位整數10(自PHP 7.4.0開始)
echo 1_234 , "<br>"; //自PHP 7.4.0 開始
```

資料型別 - 浮點數 : float, double

- 浮點數：
  - 可以帶有小數點，可以含有10的幾次方以E或e來表示。如1.23E24
- 練習：

```
echo -123.4;      //會顯示-123.4
echo +12.34       //會顯示12.34
```

資料型別 - 布林值 : boolean

- 只有兩個值true, false
- 當資料的性質只有兩種時，適合使用此資料型別。如:
  - 是、否
  - 有、無
  - 對、錯
- 另一主要用途是可用來做為決策的依據，根據這個值來決定哪些事要做，哪些事不做。
- PHP的falsy成員: false,0.0,0,"0","",null,[]

資料型別 - 字串：string(單引號)

- 文字性的資料以單引號或雙引號括起來
- 單引號：
  - 單引號字串提供[\\]和[\']兩個跳脫字元，分別被解譯成[\]和[']
- 例：

```
echo 'My name is Sara',"<br>";
echo 'My name is \'Sara\'',"<br>";
echo 'My name is "Sara"',"<br>";
echo 'My name is \\Sara\\',"<br>";
```

輸出結果

```
My name is Sara
My name is 'Sara'
My name is "Sara"
My name is \Sara\
```

資料型別 - 字串：string(雙引號)

- 雙引號字串
  - 也有提供跳脫字元[\\]和[\"]，分別被解譯成[\]和["]
- 例：

```
echo "My name is Sara","<br>";
echo "My name is 'Sara'","<br>";
echo "My name is \"Sara\"","<br>";
echo "My name is \\Sara\\","<br>";
```

輸出結果：

My name is Sara
My name is 'Sara'
My name is "Sara"
My name is \Sara\

資料型別 - 單引號與雙引號的差別

- 提供的跳脫字元群不一樣
- 單引號字串中無法套用變數替換，雙引號可以套用變數替換規則

```
<?php
$name='Alice';
echo 'Hello $name<br>';
echo "Hello $name<br>";
echo "Hello {$name}<br>";
?>
```

資料型別 - null

- 表示變數沒有值

$_SERVER變數

- 可以使用$_SERVER變數取得伺服端(Server Site)或客戶端(Client Site)的相關資訊：
  - $_SERVER["REMOTE_ADDR"] //客戶端的IP位址
  - $_SERVER["PHP_SELF"] //目前檔案路徑及名稱
  - 可從phpinfo()函數查詢各環境變數的相關資訊

表單欄位變數-表單介紹

- 表單的主要用途是讓使用者在客戶端的瀏覽器中鍵入資料，以提交資料給伺服端的程式使用。表單由許多欄位組成。

- <form method="提交方式" action="伺服端程式" >
  - 提交方式有兩種：get及post
  - 伺服端程式：指出此表單要送到哪一支伺服端的程式來處理

表單欄位變數 - 取得get送來的資料

- PHP程式收到前端以get方式送來的資料後，會將其放在$_GET[]陣列中，可用$_GET["欄位名稱"]取得從客戶端送來的資料，如
  - login.html

```
<form method="get" action="login.php">
帳號:<input type="text" name="memId" size="8" maxlength="6"><br>
密碼:<input type="password" name="memPsw" size="8" maxlength="6"><br>
<input type="submit" value="送出">
</form>
```

  - login.php

```
<?php
 echo "使用者 : " , $_GET["memId"], "<br>";
 echo "密碼 : " , $_GET["memPsw"], "<br>";
?>
```

表單欄位變數 - 取得post送來的資料

- PHP程式收到前端以post方式送來的資料後，會將其放在$_POST[]陣列中，可用$_POST["欄位名稱"]取得從客戶端送來的資料，如
  - **login.html**

```
<form method="post" action="login.php">
帳號:<input type="text" name="memId" size="8" maxlength="6"><br>
密碼:<input type="password" name="memPsw" size="8" maxlength="6"><br>
<input type="submit" value="送出">
</form>
```

  - **login.php**

```
<?php
 echo "使用者 : " , $_POST["memId"], "<br>";
 echo "密碼 : " , $_POST["memPsw"], "<br>";
?>
```

表單欄位變數 - textarea欄位

- 文字區域欄位(textarea)：
  - PHP取得自客戶端以textarea送來的資料，若直接使用echo方式送出，則資料中的換列符號並不會在網頁中以跳列方式呈現，必須使用nl2br()函數才能將換行符號\n，替換成
換列標籤。

  - **login.html,加入下列**

```
有話要說:<br>
 <textarea name="note" rows="3" cols="50"></textarea>
```

  - **login.php,加入下列程式碼**

```
<?php
echo "說了什麼(原始): ", $_GET["note"];
echo "<hr>";
echo "說了什麼(轉br): ", nl2br($_GET["note"]);
?>
```

常數

- 固定值，設定後不可再改變
- 語法：

  const 常數名稱 = 值;

- 內建常數(build-in)
  - 使用者不需事先定義，即可直接使用，如
  - PHP_VERSION、PHP_OS、M_PI

資料型別的檢查

- gettype(資料);
  - 傳回"資料型別"（ string, integer, double, array, object,.... ）
- is_....
  - 判斷是或否，會傳回布林值:true或false
  - is_integer(資料)、is_float(資料)、is_bool(資料)、is_string(資料)、is_null(資料)、is_resource(資料)、is_array(資料)、is_object(資料)、is_numeric(資料)
- var_dump(資料);
  - 顯示資料的 “資料型別"與"值"

資料型別的檢查-範例

```php
<?php
  echo gettype(123),"<br>";
  echo is_string("12345"),"<br>";
  var_dump(3.1*2, true);
?>
```

輸出結果

```
integer
1
float(6.2) bool(true)
```

資料型別的轉換_1

- 自動轉型，不需額外的程式碼
- 例：

```
$aInt = 123;
$bStr = "100";
$cBool =  true;

echo ($aInt + $bStr) , "<br>";
echo ($aInt + $cBool) , "<br>";
echo ($bStr + $cBool) , "<br>";
```

輸出結果

223

124

101

資料型別的轉換_2

- 強制轉型，需要額外的程式碼
  - 使用轉型運算式(int)(integer) (float) (double) (array)...
    - 例:

```
$a = 123.5;
$b = (int) $a;  //$a型別沒改變
var_dump($a, $b);
```

  - 使用setType(變數，型別)
    - 例:

```
$a="123.5";
settype($a, “integer”); //$a型別會改變
echo  $a, "<br>";
var_dump( $a );
```

資料型別的轉換_3

- 各種型別若轉換為布林值的規則為:
  - 以下之資料會轉換成false
    - 整數0、浮點數0.0 、空字串"" 、字串 “0"、null、沒有元素的陣列
    - 其它資料均會轉換成true

單元六：運算式

- 運算式簡介
- 算術運算子(Arithmetic Operators)
- 比較運算子(Comparison Operators)
- 邏輯運算子(Logical Operators)
- 位元運算子(Bitwise Operators)
- 字串結合運算子(string operators)
- 三元運算子(Ternary conditional operator)
- 指定運算子(Assignment Operators)

運算式簡介

- 「運算式」(Expressions)是由「運算子」(Operators)和「運算元」(Operands)組成,如:

類別	範例	結果
算術運算式	2 * 3 + 5	11
比較運算式(關係)	10 > 5	true
邏輯運算式	10 > 5 && 0 > 2	false
指定運算式	$a = 10 * 2	$a的值為20
字串結合運算式	"Name : " . "Amy"	"Name : Amy"

  - 「+」、「*」、「>」、「 &&」和「=」為運算子(Operators)
  - 數值 2、3、5..及變數$a都是運算元(Operands)

算術運算子(Arithmetic Operators)

- 用來撰寫一般的數學運算式：(假設$a=10)

運算子	說明	運算式範例	結果
- (一元運算子)	負號	-$a	-10
*	乘法	$a * 2	20
/	除法	$a / 2	5
%	餘數	$a % 2	0
+	加法	$a + 2	12
-	減法	$a – 2	8

算術運算式範例

```
<?php
echo "14*4=", 14*4 ,"<br>"; //56
echo "14/4=", 14/4 ,"<br>";  //3.5
echo "14%4=", 14%4 ,"<br>";  //2
?>
```

遞增/遞減運算子_1 (Incrementing/Decrementing Operators)

- 遞增/遞減運算子可以置於變數之前或之後
  - 放在前面，變數值先改變，再取值執行運算式
  - 放在後面，則取值執行運算再改變變數的值
- 例一：

```
<?php
$x = 10;
$y = ++$x + 5;
echo "x : $x <br>"; //11
echo "y : $y <br>"; //16
//------------
$x = 10;
$y = $x++ + 5;
echo "x : $x <br>"; //11
echo "y : $y <br>"; //15
?>
```

遞增/遞減運算子_2 (Incrementing/Decrementing Operators)

- 例二：

```
<?php
$a = 10;
$b = ++$a + ++$a;  //a: 11,12  , (11 + 12)
echo "a : $a <br>";
echo "b : $b <br>";
//------
$a = 10;
$b = $a++ + $a++; //(10 + 11) , a: 11,12
echo "a : $a <br>";
echo "b : $b <br>";
?>
```

比較運算子(Comparison Operators)

- 常被用來做為決策的依據,如在條件敘述和迴圈敘述中的判斷條件。(假設 $a=6, $b=0)

運算子	說明	運算式範例	結果
==	等於	false == $b	true
===	識別，不只值相等，型態也需相同	false === $b	false
!=	不等於	$a != $b	true
<	小於	$a < $b	false
>	大於	$a > $b	true
<=	小於等於	$a <= $b	false
>=	大於等於	$a >= $b	true

邏輯運算子(Logical Operators)

- 連接多個比較運算式以建立實際運作時的複雜運算式

運算子	範例	說明
!	! op	Not 運算，傳回 op 的相反值，true 成 false
&&	op1 && op2	And 運算，連結的 2 個運算元都為 true，結果為 true
||	op1 || op2	Or 運算，連結的 2 個運算元，任一個為 true，結果為 true
and	op1 and op2	And 運算，連結的 2 個運算元都為 true，結果為 true
or	op1 or op2	Or 運算，連結的 2 個運算元，任一個為 ture，結果為 true
xor	op1 xor op2	xor 運算，連結的 2 個運算元，其值不同為 true，其值相同為 false

位元運算子(Bitwise Operators)

- 用來執行二進位值的位元運算，如下表所示：

運算子	範例	說明
~	~op	位元的 Not 運算也就是 1’ 補數運算元，即位元值的相反值，1 成 0，0 成 1
&	op1 & op2	位元的 And 運算，2 個運算元的位元值相同是 1 時為 1，如果有一個為 0，就是 0
|	op1 | op2	位元的 Or 運算，2 個運算元的位元值只需有一個是 1，就是 1，否則為 0
^	op1 ^ op2	位元的 Xor 運算，2 個運算元的位元值只有一個為 1，結果為 1，如果同為 0 或同為 1 時結果為 0
<<	op1 << op2	op1 運算元向左位移 op2 個位元
>>	op1 >> op2	op1 運算元向右位移 op2 個位元

字串結合運算子(string operators)

- 字串結合運算子「.」可以結合2個字串成為一個字串，例：
  - "Name : " . "Amy"
  - "<h1>" . "e 拍即合網" ."</h1>"

三元運算子 (Ternary conditional operator)

- 根據某一個條件運算式的結果為true或false來決定傳回值A或值B
- 語法: 條件運算式 ？ 值A : 值B
- 例:

```
$avg = 86;
echo $avg>=60 ? "通過!" : "再努力!";
```

指定運算子(Assignment Operators)

運算子	範例	相當的運算式	說明
=	$x = $y	$x = $y	指定敘述
+=	$x += $y	$x = $x + $y	數字相加指定敘述
.=	$x .= $y	$x = $x . $y	字串連結指定敘述
-=	$x -= $y	$x = $x - $y	減法指定敘述
*=	$x *= $y	$x = $x * $y	乘法指定敘述
/=	$x /= $y	$x = $x / $y	除法指定敘述
%=	$x %= $y	$x = $x % $y	餘數指定敘述
<<=	$x <<= $y	$x = $x << $y	位元左移 y 位元指定敘述
>>=	$x >>= $y	$x = $x >> $y	位元右移 y 位元指定敘述
&=	$x &= $y	$x = $x & $y	位元 And 運算指定敘述
|=	$x |= $y	$x = $x | $y	位元 Or 運算指定敘述
^=	$x ^= $y	$x = $x ^ $y	位元 Xor 運算指定敘述

單元七：程式的流程控制

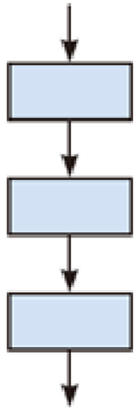
- 結構化程式設計
- 選擇結構-if句型
- 選擇結構-switch句型
- 重覆結構-for句型
- 重覆結構-while句型
- 重覆結構-do...while句型
- 巢狀迴圈

結構化程式設計

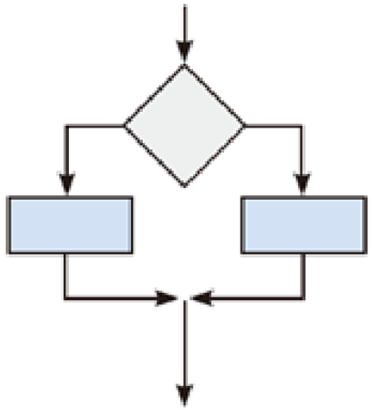
- 程式碼大部分是一列指令接著一列指令循序的被執行，但是對於現實生活中的各項複雜工作，我們還需要使用其他的「流程控制結構」才能順利的完成我們所要的工作。
- 結構化程式設計含有三種結構
  - 循序結構：指令一列接著一列循序的被執行
  - 選擇結構：根據某些條件的判斷以決定執行哪一個區塊的程式碼
  - 重覆結構：根據某些條件以決定是否要重複執行某一區塊的程式碼，又稱為迴圈結構

結構化程式設計

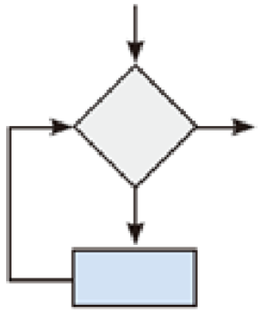
循序結構



選擇結構



重覆結構



選擇結構-if句型_1

- 句型

```
if(條件){
        敘述區塊一
}
[else{
        敘述區塊二
 }]
```

- 條件成立(true)執行述敘區塊一
- 條件不成立(false)執行述敘區塊二
- 若敘述區塊中的敘述只有一行，可以省略大括號
- 註:

  [else {
  程式區塊N
  }]
  語法介紹中的**中括弧意指此段結構不一定要在句型中出現**

選擇結構-if句型_2

- 例：

```
if($avg>=60){
   echo "成功!<br>";
   echo  “恭喜<br>";
}
else{
   echo "再努力!<br>";
   echo  “加加油!!!<br>";
}
```

選擇結構-if句型_3

- 當我們的條件無法以簡單的二分法決定流程時，可以使用下列句型

```
if (條件式一){
      程式區塊一
} elseif (條件式二){
      程式區塊二
} elseif (條件式三){
      程式區塊三
      .....
}[else {
            程式區塊N
}]
```

選擇結構-if句型_4

- 例:

```
$avg = 88;
if($avg >= 90)
   $grade = "A";
elseif($avg >= 80)
        $grade = "B";
elseif($avg >= 70)
        $grade = "C";
elseif($avg >= 60)
        $grade = "D";
else
        $grade = "E";
echo "等級為 $grade";
```

註:PHP提供了一些流程控制的替代語法

- 包括 if，while，for，foreach 和switch。替代語法的基本形式是把左花括號（{）換成冒號（:），把右花括號（}）分別換成 endif;，endwhile;，endfor;，endforeach; 以及endswitch;

選擇結構-switch句型_1(語法)

```
switch(運算式){
  case 值1:
     敘述區塊1
     [break;]
  case 值2:
     敘述區塊2
     [break;]
…
  case 值n:
     敘述區塊n
     [break;]
  [default:
     敘述區塊n+1]
}
```

```
switch(運算式)：
  case 值1:
     敘述區塊1
     [break;]
  case 值2:
     敘述區塊2
     [break;]
…
  case 值n:
     敘述區塊n
     [break;]
  [default:
     敘述區塊n+1;
endswitch;
```

選擇結構-switch句型_2(範例)

```
$lowType="1";
switch($lowType){
   case "1":
          $amt = 6000;
          break;
   case "2":
          $amt = 5000;
          break;
   case "3":
          $amt = 4000;
          break;
   default:
          $amt = 0;
}
echo "補助款為 $amt 元";
```

重覆結構-for句型_1(語法)

- 語法

```
for(初始設定;執行條件; 遞增或遞減運算式){
    敘述區塊      //重覆做一堆事
    [break;]
    [continue;]
}
```

- 註:

  若敘述區塊只有一列敘述，可以省略{}

重覆結構-for句型_2(語法說明)

- 說明：
  - 步驟

  1.變數初始設定只做一次

  2.若測試條件成立才執行敘述，不成立則結束重覆結構

  3.每次做完要重覆執行的敘述之後，加上遞增值或減去遞減值回到步驟2

  - break：可用來強制終止迴圈的執行

  - continue：迴圈內continue指令後的敘述不執行，直接執行下一迴圈

- 練習

重覆結構-while句型_1(語法)

- 語法

```
while (執行條件){
    敘述區塊
    [break;]     [continue;]
}
```

```
while (執行條件):
    敘述區塊
    [break;]     [continue;]
endwhile;
```

- 說明
  - 先檢查執行條件是否為true，
  - 若為true則執行迴圈內的敘述，執行完畢之後回到while（執行條件）處重新檢查，以決定是否重覆執行
  - 若為false則不執行迴圈內的敘述，直接跳到while句型之後去執行
  - 若敘述區塊只有一列敘述，可以省略{}

重覆結構-while句型_2(範例)

危險範例!

```
$i=1;
while($i<10){
   echo $i . "<br>";
}
```

迴圈內的敘述，不會被執行的範例

```
$i=10;
while($i<10){
   echo $i . "<br>";
   $i++;
}
```

正常範例

```
$i=1;
while($i<10){
   echo $i . "<br>";
   $i++;
}
```

重覆結構-do...while句型_1(語法)

- 語法：

```
do{
    敘述區塊
    [break;]    [continue;]
} while (執行條件);
```

- 說明：
  - 無條件先執行一次迴圈內的敘述，執行完畢之後再檢查while後的測試條件是否為true，
  - 若為true則執行迴圈內的敘述，執行完畢之後到while（執行條件）處重新檢查，以決定是否重覆執行
  - 若為false則不重覆執行迴圈內的敘述，直接跳到do...while句型之後去執行
  - 若敘述區塊只有一列敘述，可以省略{}

重覆結構-do...while句型_2(範例)

- 例:

```
$i=1;
do{
    echo $i . "<br>";
    $i++;
}while($i<11);
```

為什麼要提供三種重覆結構?

- 三種重覆結構都可以達到重覆執行的功能
- 但因為其字面的語意不同，我們可以選擇較適合的句型，來讓程式具有自我說明的能力

巢狀迴圈_1

- 巢狀迴圈是在迴圈內擁有其它迴圈
- 例如：
  - 在for迴圈擁有for、while或do/while迴圈
  - while迴圈內有for、while和do/while迴圈
- 巢狀迴圈可以有很多層，二、三層...都可以。

巢狀迴圈_2-九九乘法表為例(說明)

- 以九九乘法表為例，迴圈共有兩層，第一層迴圈執行9次，當$I為1時，第二層迴圈要執行9次($j的值，從1到9)

第一層迴圈$i 值	第二層迴圈$j 值								
1	1	2	3	4	5	6	7	8	9
2	1	2	3	4	5	6	7	8	9
3	1	2	3	4	5	6	7	8	9
……									
9	1	2	3	4	5	6	7	8	9

巢狀迴圈_3-九九乘法表為例(程式碼)

- 例: 與HTML結合之巢狀迴圈

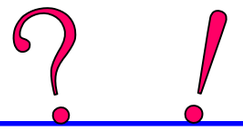
```
<table border=1 cellpadding=5>
<?php
for($i=1; $i<=9; $i++){
    echo "<tr>";
    for($j=1;$j<=9;$j++){
        echo "<td>",$i * $j,"</td>";
    }
    echo "</tr>";
}
?>
</table>
```

單元八：PHP的陣列介紹

- 陣列簡介
- 陣列類型
- 一維陣列
- 陣列的相關函數
- 二維陣列
- 表單上的陣列型資料

陣列簡介_1

- 主要功能: 有一群資料，他們所代表的意義相同，被拿來處理的過程也相近，如班上40位同學的各科成績，其被用來計算學期成績的過程是一樣的



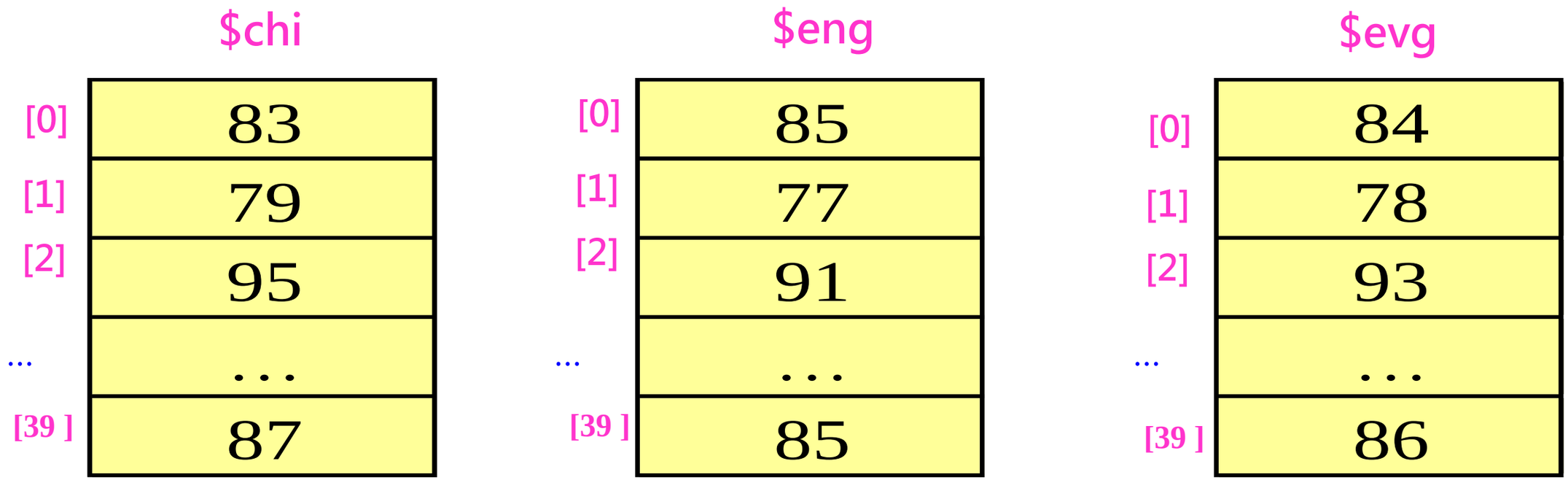
```
$chi1=83
$chi2=79
$chi3=95
…
$chi40=87
```

```
$eng1=85
$eng2=77
$eng3=91
…
$eng40=85
```

```
$avg1= ($chi1 +  $eng1)/2
$avg2=( $chi2 + $eng2)/2
$avg3= ($chi3 + $eng3)/2
…
$avg40 = ($chi40 + $eng40)/2
```

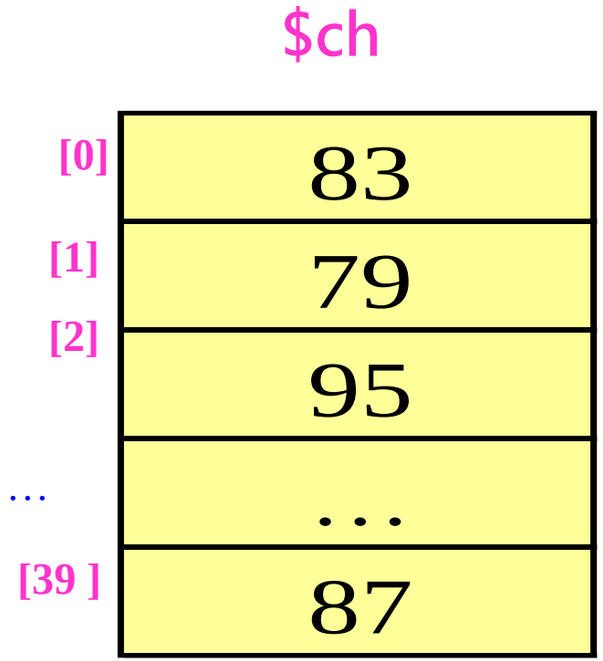
陣列簡介_2

- 以宣告陣列的方式來替我們所要處理的資料在記憶體中要出一塊區域，不只在宣告時相當的簡潔，同時也可利用迴圈的技巧來簡化我們的程式



陣列類型--依存取模式來分_1

- (一)索引陣列(Indexed Array)：
  以索引值來存取陣列中某元素的資料，索引值從0開始編起




```
echo $ch[0];
```

陣列類型--依存取模式來分_2

- (二)關聯陣列(Associative Array)：
  以使用者自訂的鍵值(key)來存取陣列中某元素的資料，鍵值可為數值或字串


```
echo $zip["中壢市"];
```

...

一維陣列-建立一維索引陣列的方法

- 直接指定

```
$ch[]=83;
$ch[]=79;
$ch[]=95;
```

```
$ch[0]=83;
$ch[1]=79;
$ch[2]=95;
```

- 使用array(資料1，資料2，資料3...)

```
$ch=array(83,79,85);
```

一維陣列-存取一維索引陣列的方法

- 使用for句型

```
for($i=0;$i <3; $i++){
    echo $ch[ $i ] "<br>";
}
```

- 使用**foreach(陣列名稱 as [索引變數=>]資料變數)**句型

```
foreach($ch as $index => $data){
    echo $data "<br>";
}
```

一維陣列-建立一維關聯陣列的方法

- 直接指定

```
$zip["台北市"]="001";
$zip["中壢市"]="320";
$zip["楊梅鎮"]="326";
```

- 使用array(鍵值1=>資料1，鍵值1=>資料2，...)

```
$zip = array("台北市"=>"001", "中壢市"=>"320 “,"楊梅鎮"=>"326");
```

一維陣列-取用一維關聯陣列的方法

- 使用foreach句型

  foreach(陣列名稱 as [鍵值變數=>]資料變數)

- 例:

```
foreach($zip as $key => $data)
  echo "鍵值 : $key --> 資料: $data<br>";
}
```

一維陣列-陣列的操作

- 個別元素的操作
  - $ch[3]=83;
- 整體陣列的操作
  - $chArr= $ch;

陣列的相關函數

- count(陣列變數):傳回陣列中元素的個數
- print_r(陣列變數):顯示陣列中的所有資料
- is_array(變數):檢測變數中的資料是陣列嗎?
- in_array(欲搜尋的值，陣列變數):傳回資料在陣列中嗎?
- array_search(欲搜尋的值，陣列變數):傳回資料在陣列中的索引值
- shuffle(陣列變數):將陣列中的資料打亂
- array_sum(陣列變數)
- array_values(陣列變數)
- ……

二維陣列

$ch[ ]

0	83
1	79
2	95
…	…
39	87

$en[ ]

0	85
1	77
2	91
…	…
39	85

$avg[ ]

0	84
1	78
2	93
…	…
39	86

將上面三個一維陣列改以一個二維陣列來存放，其中$score[2][1]內的值為91

	0	1	2
0	83	85	84
1	79	77	78
2	95	91	93
…	…	…	…
39	87	85	86

二維陣列-建立二維陣列

- 先建一維陣列,再建二維陣列

```
$row0=array(83,85);
$row1=array(79,77);
$row2=array(95,91);
$score=array($row0,$row1,$row2);
```

- 直接使用array建立二維陣列

```
$score=array(array(83,85),array(79,77),array(95,91))
```

二維陣列--存取二維陣列_1

- 使用for句型

```
<table border="1" align="center" cellspacing="0"
<?php
$score=array( array(1,2,3,4), array(11,12,13,14), array(21,22,23,24));
for( $i=0; $i<3; $i++){
  echo "<tr>";
  for( $j=0; $j<4; $j++){
    echo "<td>",$score[$i][$j],"</td>";
  }
  echo "</tr>";
}
?>
</table>
```

二維陣列--存取二維陣列_2

- 使用foreach句型

```
<table border="1" align="center" cellspacing="0">
<?php
$score=array( array(1,2,3,4), array(11,12,13,14), array(21,22,23,24));
foreach( $score as $row){
    echo "<tr>";
    foreach( $row as $data ){
        echo "<td>",$data,"</td>";
    }
    echo "</tr>";
}
?>
</table>
```

表單上的陣列型資料_1 (checkBox,select)

- 表單(arrayForm.html)如下:

```
<form action="arrayForm.php">
帳號<input type="text" name="memId" /><br />
英文能力<input type="checkBox" name="ability[]" value="read">讀
    <input type="checkBox" name="ability[]" value="listen">聽
    <input type="checkBox" name="ability[]" value="write">寫<br>
專長<br>
  <select name="specialty[]" size="3" multiple >
    <option value="VB.NET">VB.NET</option>
    <option value="SQL">SQL Server </option>
    <option value="JAVA">JAVA </option>
    <option value="Delphi">Delphi </option>
  </select>
<input type="submit" value="送出">
</form>
```

表單上的陣列型資料_2 (checkBox,select)

- PHP程式取得表單上的陣列型資料(arrayForm.php)

```
<?php
 echo "英文能力: ","<br>";
 for($i=0;$i<count($_GET["ability"]);$i++){
    echo $_GET["ability"][$i]."<br>";
 }

 echo "英文能力: ","<br>";
 for($i=0;$i<count($_GET["specialty"]);$i++){
    echo $_GET["specialty"][$i]. "<br>";
 }
?>
```

單元九：PHP的函式介紹

- 函式簡介
- 內建函式(Build-in functions)
- 自訂函式(User-defined functions)
- 傳值呼叫與傳址呼叫
- 陣列型參數
- 設定參數的預設值
- 變數的有效範圍
- 「引用檔案」函數

函式簡介

- 主要功能: 將一段具有某種特定功能的程式碼另外包裝成獨立的函式，以減少程式碼的重複性並增進程式的再用性，同時也可增加程式的可讀性並讓程式更易於維護。

- 分為兩類
  - 自訂函式(User-defined functions)
  - 內建函式(build-in functions)

內建函式(Build-in functions)

- 數學函式
- 字串函式
- 陣列函式
- 日期時間函式
- MySql資料庫函式
- 目錄管理函式
- 檔案系統函式
- 電子郵件函式
- ．．．．．

數學函式

- Generate a random integer
  - **rand ( )**
- Generate a random value via the Mersenne Twister Random Number Generator
  - **mt_rand()**
- Round fractions down
  - **floor()**
- More…

字串函式

- Get string length
  - **strlen($string)**
- Find the position of the first occurrence of a substring in a string
  - **strpos($string, $searchData)**
- Return part of a string
  - **substr($string, $start, $length)**
- Split a string by string
  - **explode($delimiter, $string)**
- Join array elements with a string
  - **implode ($glue, $pieces)**
- 中文的字串函式家族要加上mb_

日期時間函式

- Return current Unix timestamp
  - time()
- Format a local time/date
  - date($format, $timestamp)
- Get Unix timestamp for a date
  - mktime ($hour, $minute, $second, $month, $day, $year)

自訂函式(User-defined functions)

- 宣告函式

```
function 函式名稱(參數1, 參數2, 參數3 ...){
        敘述
        [return | return 傳回值]
}
```

- 呼叫函式

```
函式名稱(實際引數1 ,實際引數2,實際引數3...);
```

自定函式-範例一 (沒有參數,沒有傳回值)

- 宣告函式

```
function sayHello(){
   echo "Hello world...<br>";
   return;
}
```

- 呼叫函式

```
sayHello();
```

自定函式-範例二 (有一個參數,沒有傳回值)

- 宣告函式

```
function sayHelloToSomeone($name){
    echo "Hello " , $name ,"<br>";
}
```

- 呼叫函式

```
sayHelloToSomeone("Alice");
```

自定函式-範例三 (有兩個參數,有傳回值)

- 定義函式

```
function sum2num($a, $b){
    $c=$a+$b;
    return $c;
}
```

- 呼叫函式

```
echo " 10 + 20 = " , sum2num(10,20) ,"<br>";

$x=100; $y=200;
echo " $x + $y = " , sum2num($x,$y) ,"<br>";
```

自定函式-範例四 (傳入的參數為陣列)

- 宣告函式

```
function sumMany($arr){
    $total = 0;
     foreach($arr as $data){
        $total += $data;
     }
    return $total;
}
```

- 呼叫函式

```
$mathArr = array(60,70,80,90,100);
echo "全班總分 : " , sumMany($mathArr) ,"<br>";
```

傳值呼叫與傳址呼叫

- 傳值呼叫(call by value)
  - 呼叫端將引數的值傳給函式

- 傳址呼叫(call by reference)
  - 呼叫端將引數的位址傳給函式

傳值呼叫與傳址呼叫-範例

- 宣告函式

```
function sumByRefByValue(&$a,$b){
  $c = $a + $b;
  $a +=10;
  $b +=10;
  return $c;
}
```

- 呼叫函式

```
$x = 10;
$y = 20;
echo "$x+$y=",sumByRefByValue($x,$y), "<br>";
echo "x=$x<br>";
echo "y=$y<br>";
```

陣列型參數-傳值呼叫

- 宣告函式

```
function adjustSalary($dataArr,$amt){
    for($i=0;$i<count($dataArr);$i++){
       $dataArr[$i] += $amt;
    }
    return $dataArr;
}
```

- 呼叫函式

```
$salaryArr = array(10000,20000,30000,40000);
$salaryArr = adjustSalary($salaryArr,2000);
print_r($salaryArr);
```

陣列型參數-傳址呼叫

- 宣告函式

```
function adjustSalary_ByRef(&$dataArr,$amt){
    for($i=0;$i<count($dataArr);$i++)
     $dataArr[$i] += $amt;
}
```

- 呼叫函式

```
$salaryArr = array(10000,20000,30000,40000);
adjustSalary_ByRef($salaryArr,2000);
print_r($salaryArr);
```

設定參數的預設值

- 當宣告函式時若有設定參數的預設值,則呼叫函式時若沒有傳實際參數資料, 函式會採用參數的預設值
- 有預設值的參數必須列在後面

- 語法:

```
function 函式名稱(參數1, 參數2, 參數3=預設值 ...){
    敘述...
    [return | return 傳回值]
}
```

設定參數的預設值_範例

- 宣告函式

```
function printMark($unitName="前端工程師班") {
...
}
```

- 呼叫函式
  - 有預設值的參數,在呼叫時可以不用給實際參數值,如:

```
printMark();
printMark("CFD102");
```

變數的有效範圍

- 區域變數（Local Variables）：在函式內定義的變數，只能在函式內使用；當我們呼叫函式時，函式中的區域變數空間會被建立起來，函式結束時也會收回此空間；函式的參數也屬於區域變數。
- 全域變數（Global Variables）：程式中所有的區塊都可以使用此變數。（函式內若要使用全域變數可使用global 或$GLOBALS[“變數名稱"])
- 靜態變數（Static Variables）：一種區域變數，在函式結束執行後，此類變數的空間並不會被收回。(需以static宣告)

變數的有效範圍_全域變數範例_使用GLOBALS陣列

```
function getAmount(){
    //....
    //....
    $GLOBALS["amount"] = 100000;
}

function showAmount(){
    echo "營業額: ",$GLOBALS["amount"],"<br>";
}

getAmount();
showAmount();
```

變數的有效範圍_全域變數範例_使用global關鍵字

```
function getAmount2(){
  //....
  global $amount;
  $amount = 100000;
}

function showAmount2(){
  global $amount;
  echo "營業額: ",$amount,"<br>";
}
getAmount2();
showAmount2();
```

變數的有效範圍_靜態變數範例

```
function myStatic(){
    static $i = 0;     //靜態變數
    $i += 1;
    return $i;
}
echo "呼叫myStatic函數第一次, i : " , myStatic(), "<br>";
echo "呼叫myStatic函數第二次, i : " , myStatic(), "<br>";
```

「引用檔案」函數

- 可以使用此類函數直接在PHP程式中插入另一個檔案的PHP程式碼，副檔名可以是php或inc(通常是inc)

- 引用函數
  - include_once ("引用檔案的path/file");
  - require_once ("引用檔案的path/file");

單元十：PHP存取MySQL資料庫

- 存取MySQL資料庫簡介
- PHP存取MySQL的方法
- PDO常數
- PDO class
- PDOStatement class
- PDOException class
- 連線到資料庫管理系統
- 執行SQL命令
- 交易管理

存取MySQL資料庫簡介

- 連線MySQL伺服器
- 開啟資料庫
- 執行SQL命令
- 關閉連線
- 例：

```
C:\mysql\bin>mysql –h localhost –u root –p password: ***
mysql>use bookstore;
mysql>select * from member;
mysql>quit;
```

PHP存取MySQL的方法

- Mysql API :Mysql extention
- Mysqli API :Mysql Improved extention
- PDO API : PHP Data Objects Interface
  - 可以連結到更多種類的資料庫系統
  - 可以避免SQL injection(SQL隱碼)的攻擊，安全性較佳

加載 PDO API 和mysqli API

- 使用PDO API，必需在php.ini檔中設定要引用PDO的指令。
  - extension=pdo_mysql.dll

- 使用phpMyAdmin
  - 會用到mysqli API，必需在php.ini檔中設定要引用mysqli的指令。
    extension=php_mysqli

- php8會用到openssl API，必需在php.ini檔中設定要引用openssl的指令。

  extension=openssl

PDO參考文件

- PDO參考的連結如下:
  ht tps://www.php.net/manual/en/book.pdo.php
- 項目
  - Predefined Constants
  - PDO — The PDO class
  - PDOStatement — The PDOStatement class
  - PDOException — The PDOException class

PDO常數_1

PDO::PARAM_BOOL (integer)
Represents a boolean data type.

PDO::PARAM_NULL (integer)
Represents the SQL NULL data type.

PDO::PARAM_INT (integer)
Represents the SQL INTEGER data type.

PDO::PARAM_STR (integer)
Represents the SQL CHAR, VARCHAR, or other string data type.

PDO::PARAM_LOB (integer)
Represents the SQL large object data type.

PDO::PARAM_STMT (integer)
Represents a recordset type. Not currently supported by any drivers.

PDO::PARAM_INPUT_OUTPUT (integer)
Specifies that the parameter is an INOUT parameter for a stored procedure. You must bitwise-OR this value with an explicit PDO::PARAM_* data type.

PDO常數_2

PDO::FETCH_LAZY (integer)
Specifies that the fetch method shall return each row as an object with variable names that correspond to the column names returned in the result set. **PDO::FETCH_LAZY** creates the object variable names as they are accessed. Not valid inside PDOStatement::fetchAll().

PDO::FETCH_ASSOC (integer)
Specifies that the fetch method shall return each row as an array indexed by column name as returned in the corresponding result set. If the result set contains multiple columns with the same name, **PDO::FETCH_ASSOC** returns only a single value per column name.

PDO::FETCH_NAMED (integer)
Specifies that the fetch method shall return each row as an array indexed by column name as returned in the corresponding result set. If the result set contains multiple columns with the same name, **PDO::FETCH_NAMED** returns an array of values per column name.

PDO::FETCH_NUM (integer)
Specifies that the fetch method shall return each row as an array indexed by column number as returned in the corresponding result set, starting at column 0.

PDO::FETCH_BOTH (integer)
Specifies that the fetch method shall return each row as an array indexed by both column name and number as returned in the corresponding result set, starting at column 0.

PDO常數_3

PDO::ERRMODE_SILENT (integer)
Do not raise an error or exception if an error occurs. The developer is expected to explicitly check for errors. This is the default mode. See Errors and error handling for more information about this attribute.

PDO::ERRMODE_WARNING (integer)
Issue a PHP **E_WARNING** message if an error occurs. See Errors and error handling for more information about this attribute.

PDO::ERRMODE_EXCEPTION (integer)
Throw a PDOException if an error occurs. See Errors and error handling for more information about this attribute.

PDO::CASE_NATURAL (integer)
Leave column names as returned by the database driver.

PDO::CASE_LOWER (integer)
Force column names to lower case.

PDO::CASE_UPPER (integer)
Force column names to upper case.

PDO::NULL_NATURAL (integer)

PDO::NULL_EMPTY_STRING (integer)

PDO class_1

- 可用來和DB server做連線，透過它可執行一些DB的操作。提供下列方法：
  - public __construct (string $dsn [, string $username [, string $password [, array $options]]])
  - public bool beginTransaction (void)
  - public bool commit (void)
  - public mixed errorCode (void)
  - public array errorInfo (void)
  - public int exec (string $statement)
  - public mixed getAttribute (int $attribute)
  - public static array getAvailableDrivers (void)

PDO class_2

- public bool inTransaction (void)
- public string lastInsertId ([string $name = NULL])
- public PDOStatement prepare (string $statement [, array $driver_options = array()])
- public PDOStatement query (string $statement)
- public string quote (string $string [, int $parameter_type = PDO::PARAM_STR])
- public bool rollBack (void)
- public bool setAttribute (int $attribute , mixed $value)

PDOStatement class_1

- 我們可透過此類物件取得資料錄集合的相關訊息
- 屬性
  - readonly string $queryString;
- 方法
  - public bool bindColumn (mixed $column , mixed &$param [, int $type [, int $maxlen [, mixed$driverdata]]])
  - public bool bindParam (mixed $parameter , mixed &$variable [, int $data_type = PDO::PARAM_STR [, int $length [, mixed $driver_options]]])

PDOStatement class_2

- public bool bindValue (mixed $parameter , mixed $value [, int $data_type = PDO::PARAM_STR])
- public bool closeCursor (void)
- public int columnCount (void)
- public void debugDumpParams (void)
- public string errorCode (void)
- public array errorInfo (void)
- public bool execute ([array $input_parameters])
- public mixed fetch ([int $fetch_style [, int $cursor_orientation = PDO::FETCH_ORI_NEXT [,int $cursor_offset = 0]]])

PDOStatement class_3

- public array fetchAll ([int $fetch_style [, mixed $fetch_argument [, array $ctor_args = array()]]])
- public mixed fetchColumn ([int $column_number = 0])
- public mixed fetchObject ([string $class_name = "stdClass" [, array $ctor_args]])
- public mixed getAttribute (int $attribute)
- public array getColumnMeta (int $column)
- public bool nextRowset (void)
- public int rowCount (void)
- public bool setAttribute (int $attribute , mixed $value)
- public bool setFetchMode (int $mode)

PDOException class

- 在執行的過程中若產生一些狀況，我們可使用此類物件來取得例外的相關訊息。
- 屬性
  - public array $errorInfo ;
- 方法
  - public string Exception::getMessage (void)
  - public Exception Exception::getPrevious (void)
  - public mixed Exception::getCode (void)
  - public string Exception::getFile (void)
  - public int Exception::getLine (void)
  - public array Exception::getTrace (void)
  - public string Exception::getTraceAsString (void)
  - public string Exception::__toString (void)

連線到資料庫管理系統

- 建立PDO物件
- $pdo = new PDO($dsn, $user, $password,$options);
  - $dsn：data source name
    - 包含一些連線資訊
    - 不同的資料庫系統其所需的資訊不一
  - $user：使用者帳號
  - $password：使用者密碼
  - $options: 選項參數, 提供額外訊息
  - 若建立成功則傳回一個pdo物件的參考，失敗則傳回一個PDO Exception

PDO DSN

- 資料來源可包括
  - 前置詞: mysql:
  - host : 主機
  - port : port number
  - dbname : 資料庫名稱.
  - charset : 字元集
- 例:

```
$dsn="mysql:host=localhost;port=3306;dbname=books;charset=utf8";
```

PDO options_1

- 用來描述與資料庫連結或執行各項操作時的一些相關設定
- 以關聯性陣列的方式來表示
- 可設定的項目，如：
  - PDO::ATTR_CASE: Force column names to a specific case.
  - PDO::ATTR_ERRMODE: Error reporting.
  - PDO::ATTR_ORACLE_NULLS : NULL 和空字串的轉換
  - (預設是不會轉成NULL，也就是字串若為empty，則回傳的是字串)
  - PDO::ATTR_STRINGIFY_FETCHES: 是否將數值性資料轉成字串(目前Mysql還不支援此設定)

PDO options_2

- PDO::ATTR_TIMEOUT: 設定timeout的秒數
- (目前Mysql還不支援此設定)
- PDO::ATTR_AUTOCOMMIT (available in OCI, Firebird and MySQL): 是否自動commit (true)
- PDO::ATTR_EMULATE_PREPARES : 是否啟用模擬PDOprepared statements的功能(目前Mysql還不支援此設定)
- PDO::MYSQL_ATTR_USE_BUFFERED_QUERY: Use buffered queries.()
- PDO::ATTR_DEFAULT_FETCH_MODE: 設定存取資料的模式。(4)

PDO options_3

- 例：

```
$options = array(
PDO::ATTR_CASE =>PDO::CASE_NATURAL,
PDO::ATTR_ERRMODE=>PDO::ERRMODE_EXCEPTION);
```

透過pdo物件執行SQL命令

- 語法：
  - **$pdo->exec(SQL命令)**:用來執行不會取得result set的指令，如insert、update、delete
  - **$pdo->query(SQL命令)** :用來執行會取得result set的指令，如select
  - **$pdo->prepare(SQL命令)**：用來事先編譯好一個SQL敘述

$pdo->exec()

- 用來執行一個不會取得result set的sql指令
  - 語法：

    $pdo->exec(SQL命令)
  - 回傳值
    - 失敗：傳回false
    - 成功：傳回影響的列數

- 例：

```
$sql="update products set price=price+10";
$affectedRow = $pdo->exec($sql);
```

$pdo->query()

- 用來執行會取得result set的sql指令，如select
  - 語法：

    $pdo->query(SQL命令)
  - 回傳值
    - 失敗：傳回false
    - **成功：傳回PDOStatement物件**，我們可透過PDOstatement物件所提供的方法來取回result set中的資料列

使用PDOStatement取回result set

- 透過PDOStatement所提供的方法來取回資料錄：
  - **fetch** ([int $fetch_style [, int $cursor_orientation = PDO::FETCH_ORI_NEXT [, int $cursor_offset = 0]]])**回傳的是一維陣列**
  - **fetchObject** ([string $class_name = “stdClass” [, array $ctor_args]])**回傳的是一個物件**
  - **fetchAll** ([int $fetch_style [, mixed $fetch_argument [, array $ctor_args = array()]]])**回傳的是二維陣列**

$pdo->prepare()

- 事先編譯好一SQL個命令
  - 語法：

    $pdo->prepare(SQL命令)
  - 說明：
    - SQL命令中可以使用**未知數**來表示日後執行時需要被代入的資料
      - 未知數可使用?參數
      - 未知數可使用名稱參數
    - 要執行已事先編譯好的指令，必需
      - 將執行時所需的實際引數資料提供給事先編譯好的SQL指令中的參數
        - (使用PDOStatement物件的**bindValue()**方法)
      - 執行之
        - 使用PDOStatement物件的**execute**()方法
  - 回傳值：
    - 失敗：傳回false
    - **成功：傳回PDOStatement物件**

提供實際引數資料給編譯好的指令

- 可使用PDOStatement的下列方法
  - public bool **bindValue** (mixed $parameter , mixed $value)

prepare()方法的執行範例

- 範例一

```
$sql = "update products set price=price+ ?";
$statement = $pdo->prepare( $sql );
$statement->bindValue(1 , 100);
$statement->execute();
```

- 範例二

```
$sql = "update products set price=price+ :amount";
$statement = $pdo->prepare( $sql );
$statement->bindValue(":amount" , 100);
$statement->execute();
```

交易管理

- $pdo->beginTransaction()
- $pdo->rollback()
- $pdo->commit()

單元十一：網頁之間的資料傳遞

- 透過表單傳遞資料
- 透過url傳遞資料
- 透過客戶端的cookie傳遞資料
- 透過伺服端的session傳遞資料

透過表單傳遞資料

- <form name="prodForm" method=... action=...> ：表單
  - method:送出的方式,get或post
  - action:送到哪一台Web server上的哪一個程式
- <input type= ...> ：配合不同的type值有不同的輸入界面
  - text 、password、radio、checkbox、file、hidden...
- <textarea>...</textarea> ：文字區塊
- 列表清單及下拉式選單

透過url傳遞資料

- 例:

```
<a href="prodQueryByPsn.php?psn=1">
明細
</a>

<a href="prodDeleteByPsn.php?psn=2">
刪除
</a>
```

透過客戶端的cookie傳遞資料_1

- cookie是用來將瀏覽者的相關資料儲存在瀏覽器所在的電腦中，以便伺服端程式在未來能夠很方便的來取用相關的資訊(如個人資訊、客製化網頁、購物車)
- cookie是小餅乾的意思，意指這些儲存在客戶端電腦中的檔案尺寸都很小
- 要使用cookie必需在客戶端開啟瀏覽器的cookie設定
- 瀏覽器可以記錄的cookie量是有限制的

透過客戶端的cookie傳遞資料_2 -寫入及讀出cookie

- 寫入cookie
  - setcookie("cookie名稱", "值", 日期);
    (1)cookie名稱：為Cookie的名稱
    (2)值：為Cookie的值
    (3)日期：cookie保留的期限，在這時間之前有效，如果未設定時間，則關閉瀏覽器時Cookie即刻刪除

- 讀出cookie
  - $_COOKIE〔"cookie名稱"〕

透過伺服端的session傳遞資料_1

- PHP程式可以使用session在伺服端保留一些使用者的資料
- PHP程式若有啟動session功能，則當使用者第一次進入網站時，就會自動指定一個Session ID給使用者，直到使用者關掉瀏覽器。
- PHP程式以檔案來儲存這些session資料
- 可以註冊不定數量的session變數，這些資料可跨網頁來存取
- session會寫在哪裡和php.ini檔中session.save_path的設定有關

透過伺服端的session傳遞資料_2

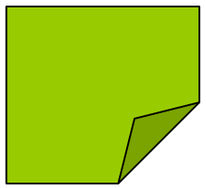
- 和session有關的函式:
  - session_start()：開始使用session
  - session_unset()：刪除所有的session變數
  - session_destroy()：刪除該session檔
  - session_id()：取得session ID
- 要使用session之前一定要先呼叫session_start()
- 啟用session之前不可送出任何資料
  - 若啟用session之前要輸出資料, 可以使用ob_start()來緩存要送出的資料
  - 目前安裝的PHP版本, 在php.ini中有啟用輸出緩存容量為4096
    - output_buffering = 4096
- 可以使用$_SESSION[“變數名稱”]來設定或取得Server端的session變數值

單元十二：上傳檔案

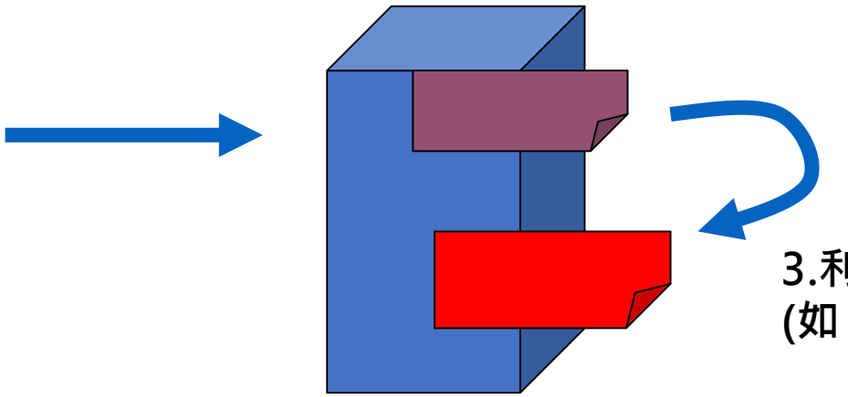
- 上傳檔案的運作模式
- Client端上傳檔案
- Server端取得上傳檔案
- 上傳多個檔案

上傳檔案的運作模式

1.Client端從表單上傳檔案至server端



2.Server端將上傳檔暫存至暫存區



3.利用copy()函數將暫存檔儲存於指定路徑
(如 "images/")

Server端允許上傳檔案的相關設定

- Server端允許客戶端上傳檔案的相關設定（php.ini)
  - file_uploads = On
  - upload_tmp_dir = "c:/wamp64/tmp"
  - upload_max_filesize = 2M

Client端上傳檔案

- 必須使用表單以**POST**方法將檔案資料上傳到伺服器
- 檔案欄位必須使用**<input type= "file" ...>**來選擇檔案
- <form>必須提供**enctype屬性指定編碼方式為multipart/form-data**，的上傳檔案模式
- 例:(fileUpload.html)

```
<form method="post" action="fileUpload.php"
  enctype="multipart/form-data">
  上傳檔案:<input type="file" name= "上傳檔案欄位"> <br>
  <input type="submit" value="送出">
</form>
```

Server端取得上傳檔案_1

- 接收上傳檔案的PHP程式可以使用$_FILES[]取得上傳檔案的相關資料
  - $_FILES["上傳檔案欄位"]["name"]：檔案名稱
  - $_FILES[“上傳檔案欄位”][“type”]：檔案種類
  - $_FILES["上傳檔案欄位"]["tmp_name"]：暫存檔名和路徑
  - $_FILES[“上傳檔案欄位”][“error”]：上傳檔案時所發生的錯誤代碼
  - $_FILES["上傳檔案欄位"]["size"]：檔案尺寸

Server端取得上傳檔案_2

- $_FILES["上傳檔案欄位"]["error"]：錯誤代碼
  - 0:上傳成功 **UPLOAD_ERR_OK**
  - 1:超過upload_max_filesize的範圍 **UPLOAD_ERR_INI_SIZE**
  - 2:超過MAX_FILE_SIZE隱藏欄位設定的範圍 **UPLOAD_ERR_FORM_SIZE**
  - 3:檔案上傳不完整 **UPLOAD_ERR_PARTIAL**
  - 4:未指定上傳的檔案 **UPLOAD_ERR_NO_FILE**
  - 6:未定義暫存區所使用的資料夾 **UPLOAD_ERR_NO_TMP_DIR**
  - 7:無法寫入暫存區 **UPLOAD_ERR_CANT_WRITE**
  - 8:php的extension停止了上傳的工作 **UPLOAD_ERR_EXTENSION**

Server端取得上傳檔案_3

- 暫存檔在PHP程式執行完畢後會被刪除，可利用copy()函數將暫存檔儲存於伺服端的指定路徑上
  - copy("來源路徑及檔名”，"複製之目的路徑及檔名")
- 例:

```
$from= $_FILES["upFile"]["tmp_name"];
$to="images/" . $_FILES["upFile"]["name"];
if(copy($from,$to))
  echo "上傳成功...";
else
  echo "上傳失敗...";
```

上傳多個檔案_1

- Client端可以同時上傳多個檔案，欄位名稱以陣列命名，如

```
<input type="file" name="檔案欄位[]"><br>
```

- PHP程式可以取得這些上傳檔案的陣列，如：

```
$name = $_FILES["檔案欄位"]["name"][$i];
$tmp = $_FILES["檔案欄位"]["tmp_name"][$i];
```

- 使用for迴圈配合copy()函數就可以將上傳的檔案放到伺服端指定的路徑

上傳多個檔案_2

- fileUploadMany.html

```
<form action="fileUploadMany.php" method="post"
  enctype="multipart/form-data">
  <fieldset style="width:300px">
    <legend>上傳檔案</legend>
    <input type="file" name="upFile[]"><br>
    <input type="file" name="upFile[]"><br>
    <input type="file" name="upFile[]"><br>
    <input type="file" name="upFile[]"><br>
  </fieldset><br>
  <input type="submit" value="送出">
</form>
```

上傳多個檔案_3

- fileUploadMany.php

```
$fileCount=count($_FILES["upFile"]["error"]);
for($i=0;$i<$fileCount;$i++)
  if($_FILES["upFile"]["size"][$i]!=0){
    $from=$_FILES["upFile"]["tmp_name"][$i];
    $to="images/".$_FILES["upFile"]["name"][$i];
    if(move_uploaded_file($from,$to))
      echo "upload : $to 成功....<br>";
    else
      echo "upload : $from 失敗....<br>";
  }
```