

# PHP + MySQL

## 網站建置基礎開發實務



資展國際 林新德

shinder.lin@gmail.com

參考專案 <https://github.com/shinder/php-basic>

# 0. 開發執行環境

🐧 安裝包： [Apache](#) + MySQL (或 [MariaDB](#)) + [PHP](#) (Apache module)

🐧 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/>)

MAMP (<https://www.mamp.info>)

🐧 Mac平台可以使用：

XAMPP ([https://www.apachefriends.org/zh\\_tw](https://www.apachefriends.org/zh_tw))，使用 ~160M 的版本

MAMP (<https://www.mamp.info>)

## 0.1 相關設定

- 埠號 (Port Number)：虛擬的對外窗口編號，一個埠號只能讓一個應用程式（服務）偵聽。
- 區域網路網段：192.168.xxx.xxx，10.xxx.xxx.xxx
- 查詢 IP 使用：
  - ▶ **ipconfig** (windows console)
  - ▶ **ifconfig** (linux, macOS X)
- Apache 的設定檔
  - C:\xampp\apache\conf\httpd.conf （主要）
  - C:\xampp\apache\conf\extra （資料夾內為附屬設定檔）
- PHP 的設定檔位置，為 phpinfo 所描述的位置
  - C:\xampp\php\php.ini

## 0.2 編輯環境

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

- 😊 PHP Intelephense ([intelephense.com](https://intelephense.com))
- 😊 PHP IntelliSense
- 😊 PHP ([devsense.com](https://devsense.com))
- 😊 Prettier - Code formatter ([prettier.io](https://prettier.io))
- 😊 indent-rainbow
- 😊 Git Graph
- 😊 Window Colors

## 0.3 建立專案

- 將 Document root 內的 index.html 修改檔名為 index\_.html。沒有預設首頁檔時，開發環境會呈現檔案資料夾列表。
- Document root 位置：[C:\xampp\htdocs](#)
- 在 Document root 建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。
- 指令 **echo**: 將資料輸出到頁面。

## 0.4 第一支 PHP 程式

```
<?php
# 儲存為 phpinfo.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  
DOCUMENT\_ROOT

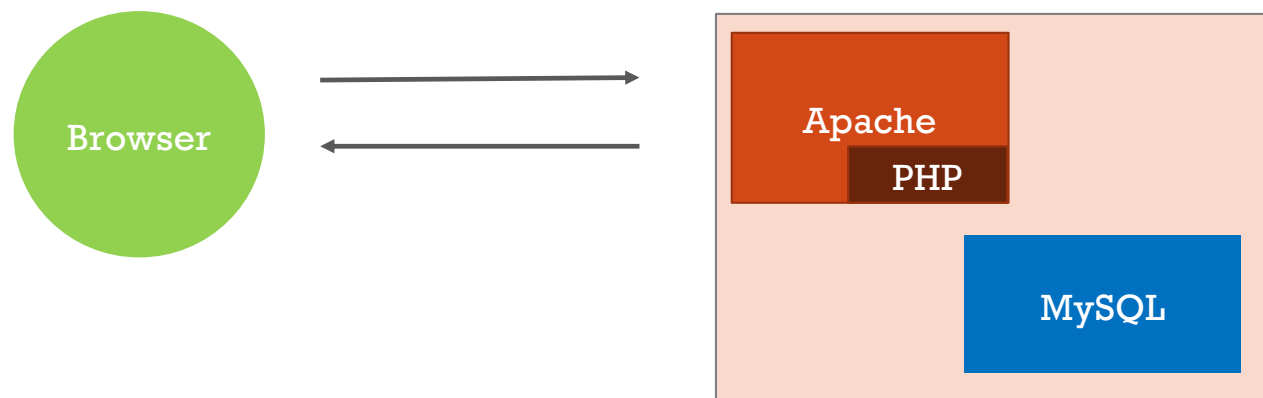
## 0.5 開啟程式錯誤顯示

```
; http://php.net/error-reporting
error_reporting=E_ALL & ~E_DEPRECATED & ~E_STRICT

; This directive controls whether or not and where PHP will output errors,
; notices and warnings too. Error output is very useful during development, but
; it could be very dangerous in production environments. Depending on the code
; which is triggering the error, sensitive information could potentially leak
; out of your application such as database usernames and passwords or worse.
; It's recommended that errors be logged on production servers rather than
; having the errors sent to STDOUT.
; Possible Values:
;   Off = Do not display any errors
;   stderr = Display errors to STDERR (affects only CGI/CLI binaries!)
;   On or stdout = Display errors to STDOUT
; Default Value: On
; Development Value: On
; Production Value: Off
; http://php.net/display-errors
display_errors=On
```

記得設定為 On (php設定檔 php.ini)

## 0.6 開發環境架構







# 1. PHP 的基本類型

- 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。

## 1.1 PHP 為弱型別程式語言

- 傳統的 PHP 變數設定時，不用宣告類型。
- 數值：int, float, double
- 布林值：bool
- 字串：string

## 1.2 常數

- 所謂常數是不可變更的值。PHP 的常數可以分成三類：一般常數或稱為純量（例：17）、系統常數（例：`PHP_VERSION`）及自訂常數。
- 一般常數（純量）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）、`__DIR__`（該 PHP 檔的所在資料夾路徑）。
- 自訂常數是程式開發者所定義的常數，舊的方式可以使用 `define()` 函式定義，較新的方式使用 `const` 關鍵字定義。

## 1.3 自訂常數

- 傳統常數使用 `define()` 定義。
- `define(常數名稱, 常數內容, 常數名稱是否不區分大小寫);`
- 亦可以使用 `const` 定義常數。

```
<?php  
const MY_PI = 3.14;  
echo MY_PI;
```

## 1.4 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; // 錯誤
```

## 1.5 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。



## 2. 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 = $a + 3`」可以簡化為「`$a += 3`」，「`+=`」為算術指定運算子。

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；

- PHP 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**。(運作方式和 JavaScript 不同)
- 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
<b>and</b>	且 (優先權比 = 低)	\$a and \$b
<b>or</b>	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。





# 3. URL 參數和基本表單

- 網址內容裡 ? 之後為 Query String 參數，也稱為 URL 參數或 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
# isset() 判斷變數有沒有設定
# intval() 把變數轉換成整數
# $_GET['a'] 參數名稱為 a 的值
$a = isset($_GET["a"]) ? intval($_GET["a"]) : 0;
echo $a;
```

- 測試網頁之後，在網址列輸入下式 ? 之後的內容。
- <http://localhost/php-basic/practices/080-query-string.php?a=25>

## ■ 3.1 基本表單和 GET 方法

```
<body>
  <!-- 沒有設定 method 屬性時，預設值為 GET -->
  <!-- 表單欄位的名稱和值會變成 query string -->
  <form action="" method="get">
    <input type="text" id="account" name="account" placeholder="帳號" />
    <br />
    <input type="password" id="password" name="password" placeholder="密碼" />
    <br />
    <input type="submit"
  </form>
  <!--
    URL ? 後面: query string parameters (或稱: GET參數, URL參數)
    格式: url-encoded
  -->
</body>
```

## ■ 3.2 基本表單和 POST 方法

```
<!-- 沒有設定 method 屬性時，預設值為 GET -->
<form name="form1" method="post" onsubmit="mySubmit(event)">
  <input type="text" name="account" placeholder="帳號">
  <br>
  <input type="password" name="password" placeholder="密碼" pattern="\d{6,}">
  <br>
  <input type="submit" value="送出">
  <br>
  <!-- *** button 如果在表單裡，type 預設為 submit -->
  <button type="button">按鈕</button>
</form>
```

```
<script>
  // e 為形式參數，接收傳入的 Event Object
  const mySubmit = function(e) {
    e.preventDefault(); // 避免預設行為（避免直接送出表單）
    // document.forms[0] // 取得表單
    // document.form1 // 表單有 name 時，取得表單
    // document.form1.elements // 表單裡所有欄位
    // document.form1.elements['account'] // 取得 account 欄位的參照
    // document.form1.account // 取得 account 欄位的參照
    // document.form1.account.value // 取得 account 欄位的值
  };
</script>
```

### ■ 3.3 Query String 的處理方式

```
<?php

$a = isset($_GET['a']) ? intval($_GET['a']) : 0;

$b = !empty($_GET['b']) ? floatval($_GET['b']) : 0;

echo $a + $b;

/*
    isset():    判斷變數或陣列成員是否有設定過
    intval():   轉換成整數
    floatval(): 轉換成浮點數
    empty():    測試裡面的值是否為 "空"
*/
```



## 4. 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 。



## 4.1 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。
- 視狀況也可以省略 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
$age = isset($_GET["age"]) ? intval($_GET["age"]) : 0;
if ($age >= 18) {
?>
    <h2>歡迎光臨</h2>
    
<?php
} else {
?>
    <h2>謝絕訪問</h2>
    
<?php } ?>
```

- PHP 提供另一種不使用大括號的語法。
- 以冒號取代左邊大括號，end?? 取代右邊大括號

```
<?php
$age = isset($_GET["age"]) ? intval($_GET["age"]) : 0;
if ($age >= 18) :
?>
    <h2>歡迎光臨</h2>
    
<?php else : ?>
    <h2>謝絕訪問</h2>
    
<?php endif; ?>
```

## 4.2 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;
}
```

## 4.3 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>
```



- 使用 sprintf() 改寫下列範列

```
<style>
* {
    font-family: "Arial";
    font-size: large;
}
table { border-collapse: collapse; }
table, tr, td { border: gray 1px solid; }
</style>
<!-- 九九乘法表 -->
<table>
    <?php for ($i = 1; $i <= 9; $i++): ?>
        <tr>
            <?php for ($k = 2; $k <= 9; $k++): ?>
                <td><?= " $k * $i = ". ($i * $k) ?></td>
            <?php endfor; ?>
        </tr>
    <?php endfor; ?>
</table>
```

```
<style>
  td {
    width: 50px;
    height: 50px;
    background-color: #000;
  }
</style>
```

```
<table>
  <?php for ($i = 0; $i < 16; $i++) : ?>
    <tr>
      <?php for ($k = 0; $k < 16; $k++) : ?>
        <td style="background-color: #<?= sprintf("%X%X0", $i, $k) ?>;"></td>
      <?php endfor ?>
    </tr>
  <?php endfor ?>
</table>
```

## 4.4 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。

## 4.5 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。

## 4.6 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;
}
```



# 5. 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。

## 5.1 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。

```
<?php

header('Content-Type: text/plain');

$ar = []; # 不需要的，但有宣告的用意
for ($i = 1; $i <= 42; $i++) {
    $ar[] = $i; # array_push()
}

shuffle($ar); # 亂數排序

// print_r($ar);
echo implode(", ", $ar); # 把陣列接成字串

# explode # 把字串切成陣列
```

- 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 />";  
}
```

## 5.2 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
header('Content-Type: text/plain');

$ar4 = [
    123,
    'name' => 'David',
    'age' => 25,
    'Hello~',
];

# 使用 foreach/as 迴圈取出陣列內容
foreach ($ar4 as $k => $v) {
    echo "$k : $v \n\n";
}
```

## 5.3 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（相對位址），必須使用 & 符號。

```
<?php
header('Content-Type: text/plain');

$ar1 = [3, 4, 5];
$ar2 = $ar1; # 設定值，相當於複製一份，不是設定參照（位址）
$ar3 = &$ar1; # 設定位址（參照）

$ar1[1] = 100;

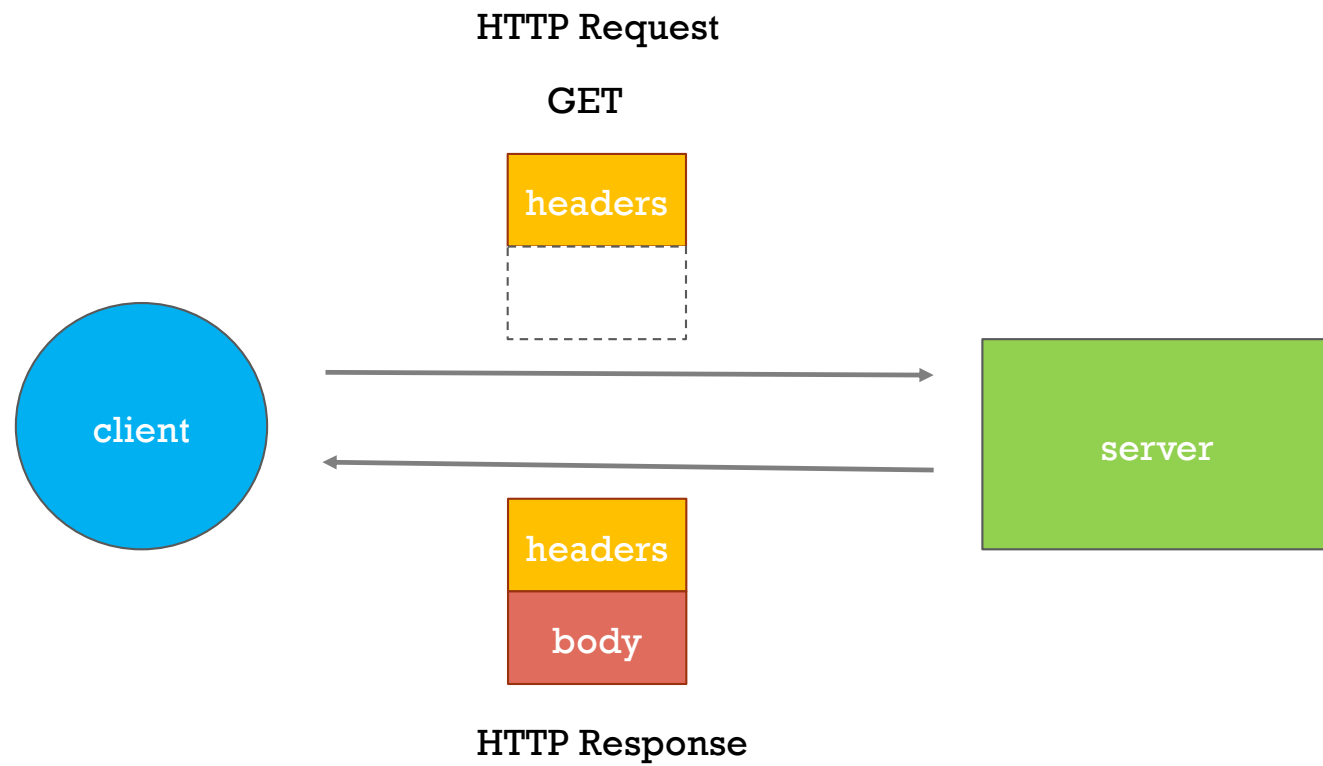
print_r($ar1);
print_r($ar2);
print_r($ar3);
```

## 5.4 \$\_GET 和 \$\_POST

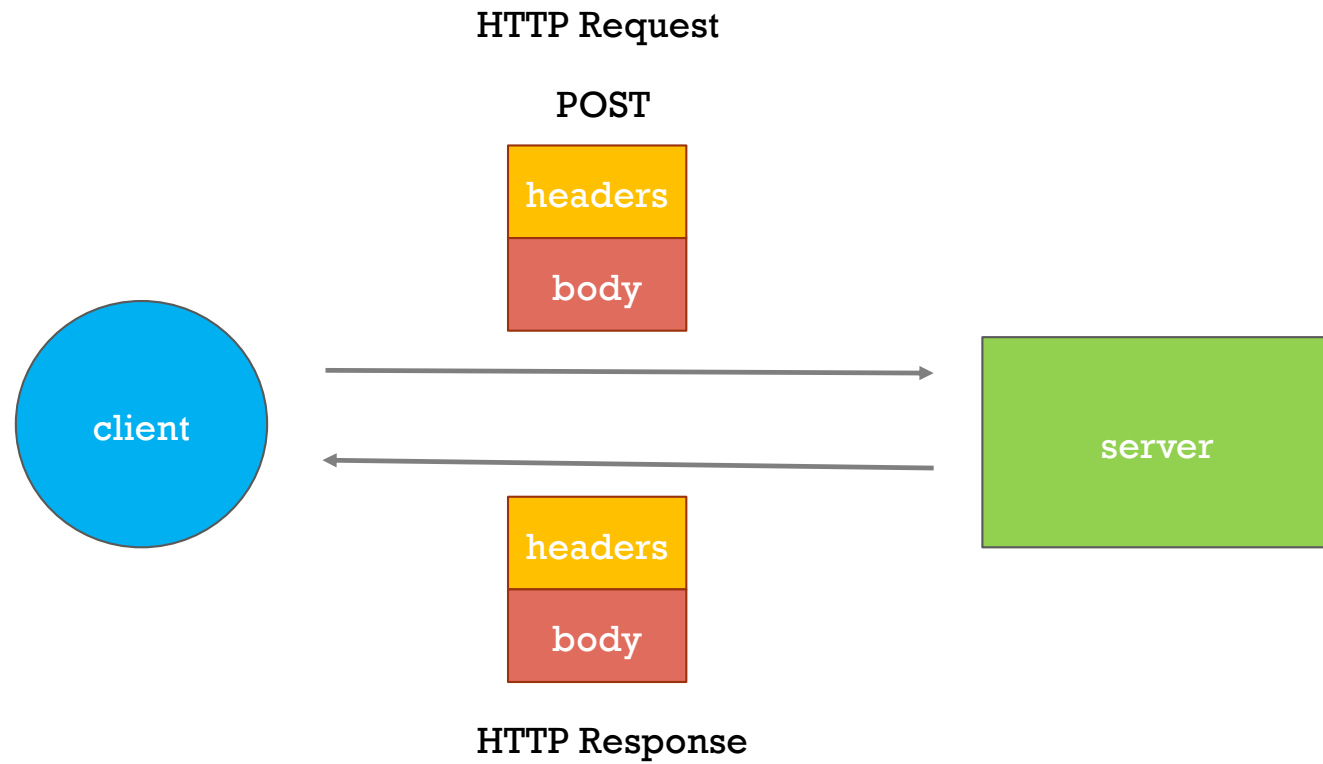
- PHP 用來取出客戶端傳來資料的 \$\_GET 和 \$\_POST 是以陣列的形式存在。
- 以下的例子在執行時，試著在網址後接著輸入「?a=123&bb=yes」，a 和 bb 會是 \$\_GET 的 key，而 123 和 yes 分別是對應的 value。
- 如何用 Chrome 觀察表單的 GET 和 POST 方法？



HTTP document



HTTP document





## 6. JSON 格式

- JSON (JavaScript Object Notation) 是一種從 JavaScript 語法而來的輕量級資料交換格式，易於人類閱讀和撰寫，也易於機器解析和生成。
- 常用於網頁應用程式中，作為伺服器 and 客戶端之間傳輸資料的格式。
- JSON 在程式語言裡一般是以字串的形式存在。
- JSON 可以表達「一個」物件 (Object)、陣列 (Array)、字串 (String)、數值 (Number)、布林值 (Boolean) 和空值 (null)。
- 字串必須使用雙引號包裹。
- 標準 JSON 文件裡不可以使用註解。

## 6.1 陣列轉換為 JSON 字串

```
<?php
header('Content-Type: application/json');

$ar1 = [
    'name' => '小新',
    'age' => 25,
    'Hello~',
];

# json_encode() 將 PHP 陣列轉換為 JSON 字串 ( JSON 的陣列格式或物件 )
# JSON_UNESCAPED_UNICODE 為內建常數，不做 unicode 跳脫
echo json_encode($ar1, JSON_UNESCAPED_UNICODE);
```

## 6.2 JSON 字串轉換為陣列

```
<?php
header('Content-Type: text/plain');

$str = '{"name":"小明", "性別":"男生"}';

$obj1 = json_decode($str); # 拿到的是 PHP 的物件 (stdClass)
var_dump($obj1);
echo $obj1->name . "\n\n";

$ar1 = json_decode($str, true); # 拿到的是 PHP 的陣列
var_dump($ar1);
```

🐛 `json_decode` 無法解析時會拿到什麼結果？



# 7. AJAX 和表單資料

- <https://zh.wikipedia.org/wiki/AJAX>
- AJAX 即「**A**synchronous **J**avaScript **a**nd **X**ML」（非同步的 JavaScript 與 XML 技術），指的是一套綜合了多項技術的瀏覽器端網頁開發技術。
- 在不刷新整個頁面的情況下，使用 JavaScript 發送要求到後端，待後端回應後更新頁面上部份的內容。
- AJAX 應用可以僅向伺服器傳送並取回必須的資料，並在客戶端以 JS 處理來自伺服器的回應，並不重新載入整個頁面。
- 目前資料交換格式已經由 XML 轉變成以 **JSON** 為主流。
- 因為在伺服器和瀏覽器之間交換的資料減少，伺服器回應相對的也更快。



## ■ 7.1 傳統的表單送出

```
<!--  
<pre> 標籤中間的「空白」字元會被忠實呈現  
-->  
<pre><?php print_r($_POST) ?></pre>  
<form name="form1" method="post">  
  <input type="text" name="account" placeholder="帳號">  
  <br>  
  <input type="password" name="password" placeholder="密碼">  
  <br>  
  <input type="submit" value="送出">  
</form>
```

```
<!--  
    預設值 enctype="application/x-www-form-urlencoded"  
    若要上傳檔案使用 enctype="multipart/form-data"  
-->  
<form name="form1" method="post" action="./240-post-data.php"  
    enctype="application/x-www-form-urlencoded">  
    <input type="text" name="account" placeholder="帳號">  
    <br>  
    <input type="password" name="password" placeholder="密碼">  
    <br>  
    <input type="submit" value="送出">  
</form>
```

```
<?php  
header('Content-Type: application/json');  
  
# 將取得的表單資料，以 JSON 格式輸出  
echo json_encode($_POST, JSON_UNESCAPED_UNICODE);
```

## ■ 7.2 撰寫後端簡單的 API

```
<?php
# 260-plus-api-get.php
header('Content-Type: application/json');

$a = isset($_GET['a']) ? intval($_GET['a']) : 0;
$b = isset($_GET['b']) ? intval($_GET['b']) : 0;

echo json_encode([
    'a' => $a,
    'b' => $b,
    'result' => $a + $b,
]);
```

```
<?php
# 270-plus-api-post.php
header('Content-Type: application/json');

$a = isset($_POST['a']) ? intval($_POST['a']) : 0;
$b = isset($_POST['b']) ? intval($_POST['b']) : 0;

echo json_encode([
    'a' => $a,
    'b' => $b,
    'result' => $a + $b,
]);
```

## ■ 7.3 使用 jQuery 的 AJAX 功能

```
<input type="number" id="aa" /> + <input type="number" id="bb" />  
<button onclick="sendData()"></button>  
<span id="result"></span>  
<script src="../../../resources/jquery-3.7.1.min.js"></script>
```

```
<script>  
  const a_field = $("#aa");  
  const b_field = $("#bb");  
  const sendData = () => {  
    const a = a_field.val();  
    const b = b_field.val();  
    // $.get(對象URL, 傳送的資料, 回呼函式, 回傳的資料格式)  
    $.get('./260-plus-api-get.php', {a, b}, function (data) {  
      if (data && data.result) {  
        $("#result").html(data.result);  
      }  
    }, 'json');  
  };  
</script>
```

```
// 以 POST 方法做 AJAX
// $.post(對象URL, 傳送的資料, 回呼函式, 回傳的資料格式)
// 送出的格式為 application/x-www-form-urlencoded
$.post('./270-plus-api-post.php', {a, b}, function (data) {
    if (data && data.result) {
        $("#result").html(data.result);
    }
}, 'json');
```

## ■ 7.4 使用 jQuery.post() 傳送整個表單資料

```
<form name="form1" onsubmit="sendData(event)">
  <input type="text" name="account" placeholder="帳號">
  <br>
  <input type="password" name="password" placeholder="密碼">
  <br>
  <input type="submit" value="送出">
</form>
<div id="result"></div>
```

```
const sendData = (e) => {
  e.preventDefault(); // 不要讓表單以傳統的方式送出
  const urlencoded = $(document.form1).serialize();
  $.post('./240-post-data.php', urlencoded, function (data) {
    $("#result").html(JSON.stringify(data));
  }, 'json');
};
```

## ■ 7.5 使用原生的 fetch() 執行 AJAX 功能

```
// fetch() 使用 GET 方法發送

fetch(`260-plus-api-get.php?a=${a}&b=${b}`)
  .then((response) => {
    if (response.ok) {
      return response.json(); // 把結果當作 JSON 字串, 做 JSON.parse() 解析
    }
  })
  .then((data) => {
    if (data && data.result) {
      document.querySelector("#result").innerHTML = data.result;
    }
  })
  .catch((ex) => {
    console.log("發生錯誤了:", ex);
  });
```



```
// fetch() 使用 POST 方法發送

fetch(`270-plus-api-post.php`, {
  method: 'POST',
  body: new URLSearchParams({a, b}),
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded'
  }
})
.then((r) => r.json()) // 把結果當作 JSON 字串，做 JSON.parse() 解析
.then((data) => {
  if (data && data.result) {
    document.querySelector("#result").innerHTML = data.result;
  }
})
.catch((ex) => {
  console.log("發生錯誤了:", ex);
});
```

## ■ 7.6 使用原生的 fetch() 傳送整個表單資料

```
const sendData = (e) => {  
  e.preventDefault(); // 不要讓表單以傳統的方式送出  
  const fd = new FormData(document.form1); // 建立沒有外觀只包含資料的表單物件  
  
  fetch(`240-post-data.php`, {  
    method: 'POST',  
    body: fd  
  })  
    .then((r) => r.json()) // 把結果當作 JSON 字串，做 JSON.parse() 解析  
    .then((data) => {  
      $("#result").html(JSON.stringify(data));  
    })  
    .catch((ex) => {  
      console.log("發生錯誤了:", ex);  
    });  
};
```

## ■ 7.7 多個欄位使用相同名稱

- 多個欄位使用相同名稱時，名稱後必需加一對中括號，表示其為陣列。
- 資料轉換成 `$_POST` 時，會變成下層的陣列。
- 若名稱後沒有中括號，則只會取得第一個值。

```
<!-- 動態增加欄位的表單 -->
<form method="post" action="./240-post-data.php">
  <div class="mb-3">
    <button type="button" class="btn btn-warning" onclick="addItem()">
      +
    </button>
  </div>
  <div class="c-container"></div>
  <button type="submit" class="btn btn-primary submit-btn">送出</button>
</form>
```

```
const c_container = $(".c-container");  
// 取得 HTML 模版 (字串) 的函式  
const itemFpl = () => {  
  return `    <div class="card-body">  
      <div class="mb-3">  
        <label class="form-label">  
          >連絡人資料  
          <button type="button" class="btn btn-danger"   
            onclick="removeItem(event)">-</button>  
        </label>  
        <input  
          type="text"  
          class="form-control"  
          name="names[]"  
          placeholder="姓名"  
        />  
      </div>  
    </div>  
  </div>`;  
};
```

```
// 加入欄位的方式
function addItem() {
  // 加入一組 card 的 HTML 片段
  c_container.append(itemFpl());
}

// 移除欄位的方式
function removeItem(event) {
  // 找到移除按鈕的參照
  const $el = $(event.target);

  // 往上找到最接近並符合條件的元素
  $el.closest(".card").remove();
}
```



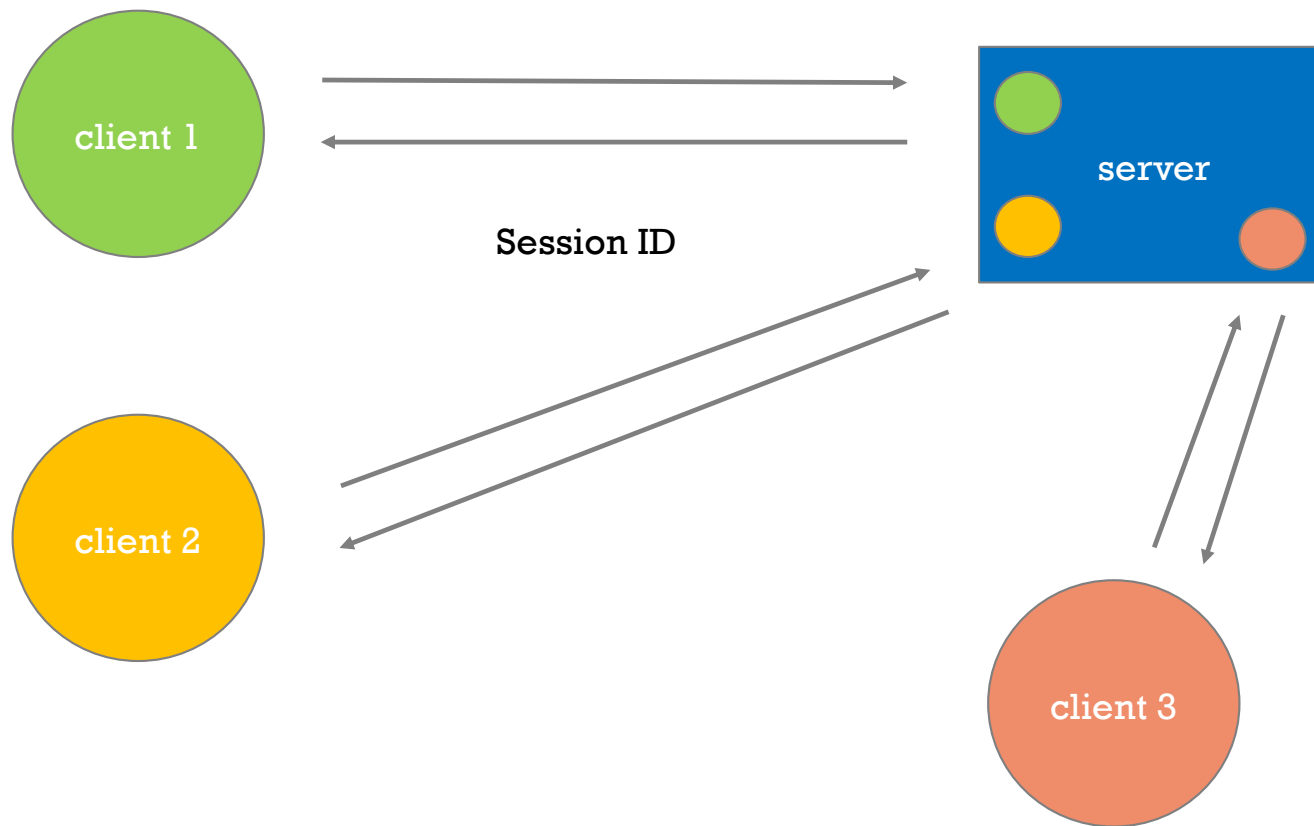
# 8. Session

- 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'];
```







## ■ 日期時間相關函式

```
<?php
# date_default_timezone_set('Asia/Taipei'); # 無法變更設定檔時，暫時變更時區

# date() 將 timestamp 轉換為一般的日期格式
printf('<h2>%s</h2>', date("Y-m-d H:i:s"));
printf('<h2>%s</h2>', date("Y-m-d H:i:s", time() + 30 * 24 * 60 * 60));
printf('<h2>%s</h2>', date("星期幾 D w", time() + 30 * 24 * 60 * 60));

# strtotime() 將日期字串轉換為 timestamp
$t = strtotime('2024/02/29');
printf('<h2>%s</h2>', date("Y-m-d", $t));

$t2 = strtotime('2023/02/29');
var_dump($t2);
printf('<h2>%s</h2>', date("Y-m-d", $t2));

$t3 = strtotime('2023/02/32');
var_dump($t3);
printf('<h2>%s</h2>', date("Y-m-d", $t3));
```

- 用戶密碼編碼 Bcrypt

`password_hash()` 將密碼轉換成雜湊碼

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

`password_verify()` 驗證雜湊碼的來源是不是該密碼

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

```
<?php
$pass = '123456';
# 將密碼轉換成雜湊碼
echo password_hash($pass, PASSWORD_DEFAULT);
```

```
<?php
$hash = '$2y$10$gi7ldq1x0FU64onECqbp10m56LQDqf0BLeN5VaG1C9Z1MsmqfvTSa';
$pass = '123456';
# 驗證雜湊碼是不是由這個密碼轉換而來
var_dump( password_verify( $pass, $hash ) );
```

- 包含檔案進來

```
<?php
# require 無法取得檔案時，會產生 error
require __DIR__ . '/350-date.php';

# include 無法取得檔案時，會產生 warning
include __DIR__ . '/350-date.php';
```



## 9. 資料庫連線 (PDO)

```
<?php
# address-book/parts/config.php 設定檔，不加入版本控制

# 資料庫連線設定
const DB_HOST = '127.0.0.1';
const DB_USER = 'root';
const DB_PASS = 'root';
const DB_NAME = 'shin01';
CONST DB_PORT = 3306;

# 設定網站常數
const PATH_BASE = '/php-basic';
```

```
<?php
# address-book/parts/db-connect.php

require __DIR__ . '/config.php';

# data source name 中間不要有空白
$dsn = sprintf('mysql:host=%s;dbname=%s;charset=utf8mb4', DB_HOST, DB_NAME);

$pdo_options = [
    # 錯誤訊息使用例外方式
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,

    # 預設 fetch 時取得索引式陣列
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
];

# 建立資料庫連線物件
$pdo = new PDO($dsn, DB_USER, DB_PASS, $pdo_options);
```



## 9.1 使用 PDO::query

```
<?php

require __DIR__ . '/parts/db-connect.php';

$sql = "SELECT * FROM address_book ORDER BY ab_id DESC LIMIT 5 ";
$stmt = $pdo->query($sql);
# $r = $stmt->fetch(); // 取得一筆資料 ( 第一筆 )
# $r = $stmt->fetch(); // 取得一筆資料 ( 第二筆 )

# 取得剩下的資料
$rows = $stmt->fetchAll(); # 使用 fetchAll() 之前，基本上不要使用 fetch()

header('Content-Type: application/json'); # 告訴瀏覽器內容為 JSON
echo json_encode($rows, JSON_UNESCAPED_UNICODE);
```

## 9.2 使用 PDO::prepare

```
<?php
require __DIR__ . '/parts/db-connect.php';

# 去掉頭尾空白, 然後轉小寫
$email = strtolower(trim($_GET['email']));

$sql = "SELECT * FROM `members` WHERE `email`=?";

# 透過連線, 將未填值的 SQL 敘述先傳給 DB 做編譯
$stmt = $pdo->prepare($sql);

# 再將值傳入執行
$stmt->execute([ $email ]);

$member = $stmt->fetch(PDO::FETCH_ASSOC);
```

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



# 10. 通訊錄列表及分頁

## 10.1 建立基本的版

```
<!DOCTYPE html>
<html lang="zh">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>a.html</title>
    <link rel="stylesheet" href="../../resources/bootstrap-5.3.3-dist/css/bootstrap.min.css" />
  </head>
  <body>
    <div class="container">
      <!-- 使用 bootstrap navbar 的範例碼 -->
      <nav class="navbar navbar-expand-lg bg-body-tertiary">...</nav>
    </div>
    <div class="container">
      <h2>a.html</h2>
    </div>
    <script src="../../resources/bootstrap-5.3.3-dist/js/bootstrap.bundle.min.js"></script>
  </body>
</html>
```

## 10.2 版型切割成數個檔案

- **html-head.php**: <body> 起始標籤之前的部份，包含整個 <head> 標籤。
- **html-navbar.php**: navbar 部份。
- **html-scripts.php**: <script> 標籤。
- **html-tail.php**: 只包含 </body> 和 </html> 兩個結束標籤。
- **init.php**: 啟動 session 和匯入 db-connect.php。

```
<?php
# address-book/parts/init.php
if (!isset($_SESSION)) {
    // 如果尚未啟動 session 的功能，就啟動
    session_start();
}
require __DIR__ . '/db-connect.php';
```

```
<!DOCTYPE html>
<html lang="zh">
<!-- address-book/parts/html-head.php -->
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title><?= empty($title) ? '小新的網站' : "$title - 小新的網站" ?></title>
  <link rel="stylesheet"
href="<?= PATH_BASE ?>/resources/bootstrap-5.3.3-dist/css/bootstrap.min.css" />
  <link rel="stylesheet"
href="<?= PATH_BASE ?>/resources/fontawesome-free-6.5.2-web/css/all.min.css" />
</head>
<body>
```

```
</body>
</html>
```

## 10.3 用戶決定查看第幾頁

```
require __DIR__ . '/parts/init.php';

$title = "通訊錄列表";
$pageName = "ab_list";

$perPage = 20; # 表示一頁最多有 20 筆
$page = isset($_GET['page']) ? intval($_GET['page']) : 1;
if ($page < 1) {
    header('Location: ?page=1'); # 跳轉頁面
    exit; # 結束程式, die()
}
```

## 10.4 條件篩選

```
$where = ' WHERE 1 ' ; # SQL 條件開頭
# 搜尋功能，搜尋兩個欄位：name, mobile
$search = $_GET['search'] ?? '' ; # ?? 的左側若為 undefined 則使用右側的值
if ($search) {
    $search_esc = $pdo->quote("%{$search}%"); # SQL 特殊字元跳脫並包裹單引號
    $where .= " AND ( `name` LIKE $search_esc OR `mobile` LIKE $search_esc ) ";
}


# 生日起始日期，轉換為 timestamp
$birthday_begin = isset($_GET['birthday_begin']) ?
    strtotime($_GET['birthday_begin']) : false;
if ($birthday_begin !== false) {
    # 轉換為單引號包起來的日期格式
    $birthday_begin = $pdo->quote(date("Y-m-d", $birthday_begin));
    $where .= " AND `birthday` >= $birthday_begin ";
}
```



## 10.5 查詢總筆數和計算總頁數

```
$t_sql = "SELECT COUNT(1) FROM address_book $where ";  
// echo $t_sql; exit; # 除錯時使用，查看 SQL 語法  
# 取得總筆數  
$totalRows = $pdo->query($t_sql)->fetch(PDO::FETCH_NUM)[0];  
# 設定預設值  
$totalPages = 0;  
$rows = [];  
if ($totalRows) {  
    # 計算總頁數  
    $totalPages = ceil($totalRows / $perPage);  
    if ($page > $totalPages) {  
        header('Location: ?page=' . $totalPages); # 跳轉頁面到最後一頁  
        exit; # 結束程式  
    }  
}  
  
# 接下頁 ➡
```

## 10.6 查詢該頁內容

```
#  呈上頁

# 取得該頁的資料
$sql = sprintf(
    "SELECT * FROM address_book %s ORDER BY ab_id DESC LIMIT %s, %s",
    $where,
    ($page - 1) * $perPage,
    $perPage
);
# 取得該頁資料
$rows = $pdo->query($sql)->fetchAll();
}
```

## 10.7 分頁按鈕 pagination

```
<ul class="pagination">
  <?php for ($i = $page - 5; $i <= $page + 5; $i++) :
    if ($i >= 1 && $i <= $totalPages) :
      # 複製 Query String 參數，並去除沒有值的參數
      $qs = array_filter($_GET);
      $qs['page'] = $i;
      # http_build_query($qs) 將陣列轉換為 urlencoded 格式
    ?>
    <li class="page-item <?= $i == $page ? 'active' : '' ?>">
      <a class="page-link" href="?<?= http_build_query($qs) ?>"><?= $i ?></a>
    </li>
  <?php
    endif;
  endfor; ?>
</ul>
```

## 10.8 資料內容以表格呈現

```
<tbody>
  <?php foreach ($rows as $r) : ?>
    <tr>
      <td><?= $r['ab_id'] ?></td>
      <td><?= $r['name'] ?></td>
      <td><?= $r['email'] ?></td>
      <td><?= $r['mobile'] ?></td>
      <td><?= $r['birthday'] ?></td>
      <td><?= htmlentities($r['address']) ?></td>
    </tr>
  <?php endforeach; ?>
</tbody>
```

## 10.9 Navbar 標示目前頁面

```
<!-- address-book/parts/html-navbar.php -->
```

```
<style>
```

```
  nav.navbar ul.navbar-nav a.nav-link.active {  
    background-color: blue;  
    color: white;  
    font-weight: 900;  
    border-radius: 6px;  
  }
```

```
</style>
```

```
<ul class="navbar-nav me-auto mb-2 mb-lg-0">
```

```
  <li class="nav-item">
```

```
    <a class="nav-link <?= $pageName == 'ab_list' ? 'active' : '' ?>"  
      href="index_.php">列表</a>
```

```
  </li>
```

```
  <li class="nav-item">
```

```
    <a class="nav-link <?= $pageName == 'ab_add' ? 'active' : '' ?>"  
      href="add.php">新增通訊錄</a>
```

```
  </li>
```

```
</ul>
```



# 11. 新增通訊錄表單及功能

## 11.1 表單頁 title 和 pageName 設定

```
<?php
# address-book/add.php

require __DIR__ . '/parts/init.php';
$title = "新增通訊錄";
$pageName = "ab_add";
?>
<?php include __DIR__ . "/parts/html-head.php"; ?>
<style>
    form .mb-3 .form-text {
        color: red;
    }
</style>
```

## 11.2 表單頁 HTML 架構

```
<h5 class="card-title">新增通訊錄</h5>
<form name="form1" onsubmit="sendData(event)" novalidate>
  <div class="mb-3">
    <label for="name" class="form-label">姓名</label>
    <input type="text" class="form-control" name="name" id="name" required>
    <div class="form-text"></div>
  </div>
  <!-- 略 -->
  <div class="mb-3">
    <label for="birthday" class="form-label">生日</label>
    <input type="date" class="form-control" name="birthday" id="birthday">
  </div>
  <div class="mb-3">
    <label for="address" class="form-label">地址</label>
    <textarea class="form-control" name="address" id="address" rows="3"></textarea>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```



## 11.3 JavaScript 檢查 email 格式

```
// http://stackoverflow.com/questions/46155/validate-email-address-in-javascript  
  
function validateEmail(email) {  
    const re =  
    /^(([^<>()\\[\]\\. ,;: \s@"]+(\.[^<>()\\[\]\\. ,;: \s@"]+)*)|(".+"))@((\[0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3}\. [0-9]{1,3})|((\ [a-zA-Z\d-]{1,4})+\.[a-zA-Z\d-]{2,}))$/;  
    return re.test(email);  
}
```

## 11.4 表單發送前的表單檢查

```
// 重置錯誤訊息
nameField.nextElementSibling.innerHTML = '';
nameField.style.border = '1px solid #CCC';
emailField.nextElementSibling.innerHTML = '';
emailField.style.border = '1px solid #CCC';
let isPass = true; // 表單有沒有通過檢查
// 表單欄位的資料檢查
if (nameField.value.length < 2) {
    isPass = false;
    nameField.nextElementSibling.innerHTML = '請填寫正確的姓名';
    nameField.style.border = '1px solid red';
}
if (!validateEmail(emailField.value)) {
    isPass = false;
    emailField.nextElementSibling.innerHTML = '請填寫正確的 Email';
    emailField.style.border = '1px solid red';
}
```

## 11.5 發送表單資料

```
if (isPass) {  
    // 將表單資料轉換為 urlencoded 格式  
    const fd = $(document.form1).serialize();  
    $.post('add-api.php', fd, function(data){  
        console.log(data);  
        if (data.success) {  
            modalBody.innerHTML = `  
                <div class="alert alert-success" role="alert">  
                    新增成功</div>`;   
        } else {  
            modalBody.innerHTML = `  
                <div class="alert alert-danger" role="alert">  
                    沒有新增</div>`;   
        }  
        modal.show();  
    }, 'json')  
}
```

## 11.6 後端欄位檢查

# address-book/add-api.php

```
require __DIR__ . '/parts/init.php';
header('Content-Type: application/json');

$output = [
    'success' => false,
    'bodyData' => $_POST, # 除錯用
];

# 表單欄位的資料檢查
# filter_var() 可以用來檢查 email 格式
# filter_var('bob@example.com', FILTER_VALIDATE_EMAIL)

$birthday = $_POST['birthday'];
$ts = strtotime($birthday); # 轉換成 timestamp
if ($ts === false) {
    $birthday = null; # 如果不是日期的格式, 就使用 null
} else {
    $birthday = date('Y-m-d', $ts);
}
```

## 11.7 後端寫入資料表

```
$sql = "INSERT INTO `address_book`(  
    `name`, `email`, `mobile`,  
    `birthday`, `address`, `created_at`  
    ) VALUES (  
        ?, ?, ?,  
        ?, ?, NOW()  
    )";
```

```
$stmt = $pdo->prepare($sql); # 準備 sql 語法, 除了 "值" 語法要合法  
$stmt->execute([  
    $_POST['name'],  
    $_POST['email'],  
    $_POST['mobile'],  
    $birthday,  
    $_POST['address'],  
]);
```

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

```
$output['success'] = !!$stmt->rowCount();  
echo json_encode($output);
```

## 11.8 列表頁避免 XSS 攻擊

- `strip_tags()` 可以移除字串中所有的標籤文字。
- `htmlentities()` 針對文字內容做 **HTML** 跳脫。

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

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



# 12. 刪除通訊錄項目

## 12.1 列表頁加入一欄

```
<?php foreach ($rows as $r) : ?>
    <tr>
        <td>
            <a href="javascript: deleteOne(<?= $r['ab_id'] ?>)">
                <i class="fa-solid fa-trash"></i>
            </a>
        </td>
        <!-- 略 -->
    </td>
</tr>
<?php endforeach; ?>
```



## 12.2 列表頁跳轉到刪除頁的 JavaScript

```
<script>
  const deleteOne = (ab_id) => {
    if (confirm(`是否要刪除編號為 ${ab_id} 的資料??`)) {
      location.href = `del.php?ab_id=${ab_id}`;
    }
  };
</script>
```

## 12.3 後端移除資料

```
# 檔案: address-book/del.php
require __DIR__ . '/parts/init.php';

$ab_id = isset($_GET['ab_id']) ? intval($_GET['ab_id']) : 0;
if (!empty($ab_id)) {
    $sql = "DELETE FROM address_book WHERE ab_id=$ab_id";
    $pdo->query($sql);
}
$come_from = "index_.php";
# 如果有 referer 的 url, 就使用 referer url
if (isset($_SERVER['HTTP_REFERER'])) {
    $come_from = $_SERVER['HTTP_REFERER'];
}

header('Location: ' . $come_from);
```



## 13. 修改通訊錄項目

- 修改資料的表單頁 ( `edit.php` ) 架構和新增資料的表單頁 ( `add.php` ) 相似。
- 主要差別在於 `edit.php` 一開始要先取得欲編輯的資料項目。

## 13.1 修改的表單頁先取得要編輯的項目

```
# 檔案: address-book/edit.php
require __DIR__ . '/parts/init.php';

$ab_id = isset($_GET['ab_id']) ? intval($_GET['ab_id']) : 0;
# 如果沒有資料編號
if (empty($ab_id)) {
    header('Location: index.php');
    exit;
}
$sql = "SELECT * FROM address_book WHERE ab_id=$ab_id";
$r = $pdo->query($sql)->fetch();
# 如果沒有該筆資料
if (empty($r)) {
    header('Location: index.php');
    exit;
}
# header('Content-Type: application/json'); # 告訴瀏覽器內容為 JSON
# echo json_encode($r);
```

## 13.2 欄位資料檢查

```
# 檔案: address-book/edit-api.php
require __DIR__ . '/parts/init.php';
header('Content-Type: application/json');
$output = [
    'success' => false,
    'bodyData' => $_POST, # 除錯用
    'code' => 0, # 除錯用
];
// 表單欄位的資料檢查
$ab_id = isset($_POST['ab_id']) ? intval($_POST['ab_id']) : 0;
if (empty($ab_id)) {
    $output['code'] = 400;
    echo json_encode($output); exit;
}
$name = $_POST['name'] ?? ''; # ?? 如果 ?? 的左邊為 undefined, 就使用右邊的值
if (mb_strlen($name) < 2) {
    $output['code'] = 405;
    echo json_encode($output); exit;
}
```

## 13.3 修改資料表項目

```
$sql = "UPDATE `address_book` SET  
    `name`=?,  
    `email`=?,  
    `mobile`=?,  
    `birthday`=?,  
    `address`=?  
    WHERE `ab_id`=? ";
```

```
$stmt = $pdo->prepare($sql); # 準備 sql 語法, 除了 "值" 語法要合法  
$stmt->execute([  
    $name,  
    $_POST['email'],  
    $_POST['mobile'],  
    $birthday,  
    $_POST['address'],  
    $ab_id,  
]);  
$output['success'] = !!$stmt->rowCount();  
echo json_encode($output);
```





# 14. 登入與權限控制

- 登入的表單頁 ( `login.php` ) 架構和新增資料的表單頁 ( `add.php` ) 相似。
- 權限的狀態使用 `session` 來記錄。

## 14.1 檢查必要欄位

```
# 檔案: address-book/login-api.php
require __DIR__ . '/parts/init.php';
header('Content-Type: application/json');

$output = [
    'success' => false,
    'bodyData' => $_POST, # 除錯用
    'code' => 401, # 除錯用
    'error' => '',
];

if (!isset($_POST['email']) or !isset($_POST['password'])) {
    $output['error'] = '欄位資料不足';
    echo json_encode($output);
    exit;
}
# 去除頭尾空白字元
$email = trim($_POST['email']);
$password = trim($_POST['password']);
```

## 14.2 檢查帳號是否正確

```
# 先判斷帳號對不對
$sql = "SELECT * FROM members WHERE email=?";
$stmt = $pdo->prepare($sql);
$stmt->execute([$email]);
$row = $stmt->fetch();

if (empty($row)) {
    $output['error'] = '帳號是錯的';
    $output['code'] = 403;
    echo json_encode($output);
    exit;
}
```

## 14.3 檢查密碼是否正確

```
# 判斷密碼對不對
if (password_verify($password, $row['password_hash'])) {
    $output['success'] = true;
    $output['code'] = 200;
    // 已登入的狀態記錄在 session 裡
    $_SESSION['admin'] = [
        'id' => $row['member_id'],
        'email' => $row['email'],
        'nickname' => $row['nickname'],
    ];
} else {
    $output['error'] = '密碼是錯的';
    $output['code'] = 405;
}

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

## 14.4 登出功能

```
<?php
# 檔案: address-book/logout.php
session_start();

unset($_SESSION["admin"]);

header('Location: index_.php');
```

## 14.5 需 admin 權限的頁面必須引入 admin-required.php

```
<?php
# 檔案: address-book/parts/admin-required.php

if (!isset($_SESSION)) {
    // 如果尚未啟動 session 的功能，就啟動
    session_start();
}

if (!isset($_SESSION["admin"])) {
    # 如果沒有登入管理者，就跳到登入頁，結束程式
    header('Location: login.php');
    exit;
}
```



# THANK YOU

