[**版本紀錄**](#_1fob9te) **2**

[**命名原則\_Naming Rules**](#_2et92p0) **3**

[檔案命名\_File names](#_tyjcwt) 3

[**寫作規範\_Coding guideline**](#_1t3h5sf) **4**

[寫作規範以 PSR-1/PSR-12 為基礎。](#_4d34og8) 4

[物件屬性 (attributes)](#_2s8eyo1) 4

[全域變數 (global variables)](#_3rdcrjn) 5

[區域變數 (local variables)](#_26in1rg) 5

[函式 (functions)](#_lnxbz9) 6

[變數與屬性的指定 (assignments of variables and attributes)](#_35nkun2) 6

[運算元 (operators)](#_1ksv4uv) if

[常數 (constants)](#_44sinio) 8

[縮寫 (abbreviations)](#_2jxsxqh) 8

[自定義設定檔 (config)](#_z337ya) 9

[翻譯檔](#_3j2qqm3) 9

[回傳 Return](#_1y810tw) 9

[函示順序](#_4i7ojhp) 9

[API 回傳](#_2xcytpi) 9

[**註解\_Annotation**](#_1ci93xb) **10**

[檔案表頭註解](#_3whwml4) 10

[類別註解](#_qsh70q) 11

[屬性與變數註解](#_3as4poj) 11

[常數註解](#_1pxezwc) 12

[方法 (methods) 與函式 (functions) 註解](#_49x2ik5) 12

[**格式\_Formatting**](#_147n2zr) **15**

[透過 PHPSTORM Code Style 設定](#_3o7alnk) 15

[透過PHP Code Sniffer 檢查規範](#_23ckvvd) 15

[**參考**](#_ihv636) **16**

[PHP Standards Recommendations](#_32hioqz) 16

# 

# 

# 版本紀錄

|  |  |  |
| --- | --- | --- |
| 日期 | 作者 | 修改項目 |
| 20190530 | Cyril\_Wang | 初始版本 |
| 20200630 | Penny\_Chen | Psr-2 改 Psr-12、增加命名原則、增加規範。 |

# 

# 命名原則\_Naming Rules

### 檔案命名\_File names

* 1. 所有名稱需為有意義的，避免過度縮寫。
  2. 大駝峰（PascalCase），無底線，不可過度縮寫。

|  |  |
| --- | --- |
| Controller | 類別名稱一律採用單數  Ex: MemberController、CategoryController |
| Model | Model 使用英文單數/首字大寫/大駝峰式命名  Ex: Member、Category |
| Seeder | ModelName + TableSeeder  Ex: UserTableSeeder、ArticleCategoryTableSeeder |
| Migration | 動詞 + Table Name + \_table  Ex: create\_users\_table、alter\_users\_table |
| Table | 使用英文複數/小寫/單字間用蛇底式命名(Snake Case)  Ex: users、roles\_has\_authorities |
| Routes | 依 RESTful 寫法，要操作的對象一律使用複數  Ex: Route::resource('members', 'MemberController'); |
| App | app 下的**目錄**命名遵守以下幾點   * 1.遵守 PSR-4 規則 * 2.單複數是依照其中的檔案是否為集合來命名；   如可能會有多個 repository ，就一律放在 Repositories 目錄下。  app 下的**檔案**命名遵守   * 使用英文單數/首字大寫/大駝峰式命名   Ex: UseService、ArticleCategoryService |
| Views | resources 下的所有目錄命名遵守以下幾點：  1.一律小寫，單字間以 dash (-) 分隔。  2.單複數依照其中的檔案是否為集合來命名；  如可能會有多種 layout ，就一律放在 resources/views/layouts 目錄下。 |

### 

# 寫作規範\_Coding guideline

寫作規範以 [PSR-1](https://www.php-fig.org/psr/psr-1/)/[PSR-12](https://www.php-fig.org/psr/psr-12/) 為基礎，另補上 PSR-1/PSR-12 未規範的事項。除非有"應盡量"字樣，否則所有規範皆須予以遵守。

### 寫作規範以 [PSR-1](https://www.php-fig.org/psr/psr-1/)/[PSR-12](https://www.php-fig.org/psr/psr-12/) 為基礎。

提醒: **new 時始終存在 ()**

|  |
| --- |
| <?php  class A  {  }  new A; *// 錯誤的*  new A(); *// 正確的* |

### 物件屬性 (attributes)

* 1. 屬性名稱使用底線 (underscore) 連接英文單字，如 i\_am\_a\_property。

|  |
| --- |
| <?php  class Foo  {  ...  // 屬性名稱使用底線 (underscore) 連接英文單字  public $i\_am\_a\_property = 'Hello World'; ...  } |

* 1. 不可在變數或屬性名稱加上表示型別的前綴字。

|  |
| --- |
| <?php  ... // 變數名稱不可加上前綴字 i  ~~$i\_count = 10;~~ ... |

### 

### 全域變數 (global variables)

* 1. 全域變數名稱使用底線 (underscore) 連接英文單字，如 welcome\_message。

|  |
| --- |
| <?php  ... // 全域變數名稱使用底線 (underscore) 連接英文單字  $welcome\_message = 'Hello World'; ... |

* 1. 不可在全域變數名稱加上表示型別的前綴字。

|  |
| --- |
| <?php  ... // 全域變數名稱不可加上前綴字 i  ~~$i\_count = 10;~~ ... |

### 區域變數 (local variables)

* 1. 區域變數名稱使用底線 (underscore) 連接英文單字，如 welcome\_message。

|  |
| --- |
| <?php  ... function showMessage()  {  ...  // 區域變數名稱使用底線 (underscore) 連接英文單字  $welcome\_message = 'Hello World';  ...  } ... |

* 1. 不可在區域變數名稱加上表示型別的前綴字。

|  |
| --- |
| <?php  ... // 區域變數名稱不可加上前綴字 i  ~~$i\_count = 10;~~ ... |

### 函式 (functions)

* 1. 函式名稱使用小駝峰的命名方式，如 doSomething。

|  |
| --- |
| <?php  ...  // 函式名稱使用小駝峰的命名方式  function doSomething() {  ... }  ... |

* 1. 開始括弧必須緊接在函式定義的下一行，而結尾括弧必須緊接在函式主題的下一行。

|  |
| --- |
| <?php  ... // 開始括弧緊接函式定義的下一行，結尾括弧緊接在函式主題的下一行  function doSomething()  {  print("Hello World"); }  ... |

* 1. 參數規範請參考 PSR-12 4.5 [Method and Function Arguments](https://www.php-fig.org/psr/psr-12/#45-method-and-function-arguments) 的規範。
  2. 函式名稱**應盡量**以動詞開頭，如 doSomething。

|  |
| --- |
| <?php  ... // 函式名稱以動詞 do 開頭  function doSomething()  {  print("Hello World"); }  ... |

### 變數與屬性的指定 (assignments of variables and attributes)

* 1. 等號 (=) 前後需各保留一個 (半形的) 空格。

|  |
| --- |
| <?php  ... // 變數指定等號 (=) 前後必須各一個空格  $i\_am\_a\_global\_variable = 'Hello World'; ... |

|  |
| --- |
| <?php  class Foo  {  ...  // 屬性指定等號 (=) 前後必須各一個空格  public $i\_am\_a\_property = 'Hello World'; ...  } |

* 1. 如換行時等號 (=) 應為最後一個字元，第二行不需保留空格，但須進行縮排。

|  |
| --- |
| <?php  ... // 等號 (=) 前方有空格，後方則無。'Hello World' 前方無空格，但須縮排。  $i\_am\_a\_global\_variable =  'Hello World'; ... |

|  |
| --- |
| <?php  class Foo  {  ...  // 等號 (=) 前方有空格，後方則無。'Hello World' 前方無空格，但須縮排。  public $i\_am\_a\_property =  'Hello World'; ...  } |

* 1. 變數與屬性的命名 (naming of variables and attributes): 不可在變數或屬性名稱加上表示型別的前綴字。

|  |
| --- |
| <?php  ... // 變數名稱不可加上前綴字 i  ~~$i\_count = 10;~~ ... |

|  |
| --- |
| <?php  class Foo  {  ...  // 屬性名稱不可加上前綴字 s  ~~public $s\_message = 'Hello World";~~ ...  } |

### 運算元 (operators)

* 1. 運算元前後需各保留一個(半形的) 空格。

|  |
| --- |
| <?php  ...  // + 前後各一個空格  $sum = $price + $tax;  ... |

* 1. 如剛好換行時運算元應為最後一個字元，第二行不需保留空格，但須進行縮排。

|  |
| --- |
| <?php  ...  // + 號前方有空格，後方則無。$tax 前方無空格，但須縮排。  $sum = $price +  $tax;  ... |

### 常數 (constants)

* 1. 常數名稱應全大寫，並以底線 (underscore) 分隔不同單字。如 APPLICATION\_NAME。

|  |
| --- |
| <?php  ... // 常數名稱全大寫，並以底線 (underscore) 分隔不同單字。  const APPLICATION\_NAME = 'My Company ERP';  ... |

|  |
| --- |
| <?php  ... // 常數名稱全大寫，並以底線 (underscore) 分隔不同單字。  define('APPLICATION\_NAME', 'My Company ERP');  ... |

### 縮寫 (abbreviations)

* 1. 除非是一般情況下也很常見的縮寫方式，否則**應盡量**避免使用縮寫。例如應避免使用 cnt 代表 counter。

### 自定義設定檔 (config)

* 1. 鍵值使用底線 (underscore) 連接英文單字。

|  |
| --- |
| <?php  return [  'i\_am\_a\_config' => 1  ]; |

### 翻譯檔

* 1. 鍵值使用底線 (underscore) 連接英文單字。

|  |
| --- |
| <?php  Return [  'i\_am\_langx' => “我是語系”  ]; |

### 回傳 Return

* 1. Return 前，必需空一行。

|  |
| --- |
| <?php  ...  $result = [  'i\_am\_result' => “abc”  ];  return $result; |

### 函示順序

**應盡量** Public 在前/ protected 在中/ private 在後。

### API 回傳

* 1. 回傳參數名稱，**應盡量**為小駝峰。

# 註解\_Annotation

### 檔案表頭註解

* 1. 一般檔案:

於檔案最上方(<?php標頭 下一行)撰寫該程式檔案用途與說明，必須至少使用下列標籤。其餘 tag 請依據該檔案處理內容、引用來進行使用：

|  |
| --- |
| <?php  /\*\*  \* 訂單管理　Controller  \* php version 7.3  \*  \* @category Controller  \* @package App\Http\Controllers  \* @author Cyril <[cyril\_wang@company.com](mailto:cyril_wang@xi-shi.net)>  \* @copyrignt 2019 My Company  \* @version GIT: @1.0.0@  \*/ |

* 1. api 檔案:

若該檔案程式處理之功能內容作為api方式提供使用，於檔案最上方除一般規範之標籤外，需另外加上標籤 @api 。其餘 tag 請依據該檔案處理內容、引用來進行使用：

|  |
| --- |
| <?php  /\*\*  \* Log紀錄處理 API  \*  \* @category Helper  \* @package App\Helpers  \* @author Cyril <[cyril\_wang@company.com](mailto:cyril_wang@xi-shi.net)>  \* @copyrignt 2019 My Company  \* @version GIT: @1.0.0@  \* @api  \*/ |

### 

### 類別註解

類別檔描述必須至少使用下列標籤，其中主旨與詳細說明 (非必要) 之間請空一行，格式如下：

|  |
| --- |
| /\*\*  \* 類別說明主旨  \*  \* 詳細說明: <description>  \*  \* @category Helper  \* @package App\Helpers  \* @author Cyril <[cyril\_wang@company.com](mailto:cyril_wang@xi-shi.net)>  \*/ |

### 屬性與變數註解

針對屬性與變數宣告，一定要使用 tag @var ，並註解其型態與說明。可使用多行或一行註解：

**格式**

|  |
| --- |
| @var [“Type”] [$element\_name] [<description>] |

**範例**

|  |
| --- |
| <?php  ...  class Captcha  {  /\*\*  \* @var int 產生圖片寬  \*/  private static $image\_width = 120;  /\*\* @var int 產生圖片高 \*/  private static $image\_height = 30;  }  ... |

### 常數註解

針對常數宣告，僅需詳細說明其用途。可使用多行或一行註解：

**範例**

|  |
| --- |
| <?php  ...  class Captcha  {  /\*\*  \* 驗證碼產生字元組合(純數字)  \*/  const TYPE\_NUMBER = '0123456789';  /\*\* 驗證碼產生字元組合(純英文) \*/  const TYPE\_WORD = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';  /\*\*  \* 驗證碼產生字元組合(數字+英文)  \*/  define('TYPE\_ANY', '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ');  } |

### 方法 (methods) 與函式 (functions) 註解

撰寫方法(或函式)時，一定要使用 @return (回傳型態)；如果方法(或函式)有傳入參數，則每個參數都必須依序搭配 tag @param (描述傳入參數型態與說明)；若在該方法(或函式)裡有可能會拋出 Exception 錯誤時，請記得於註解區塊中使用tag @throws 說明exception處理物件類別為何。

註解範例

|  |
| --- |
| <php  ...  /\*\*  \* 審核霸王金贈點  \*  \* @param int $dispense\_id 贈點紀錄 ID  \* @param string $ip 後台管理者連線 IP 位址  \* @return bool 審核處理結果  \* @throws OverlordPointAccountInsufficientBalanceException  \*/  public function approveDispense(int $dispense\_id, string $ip)  {  ...  } |

### 

return 回傳格式

1. 回傳值為明確型態

例如：

|  |  |
| --- | --- |
| @return string | 回傳字串 |
| @return int | 回傳整數 |
| @return bool | 回傳布林值 |
| @return array | 回傳陣列 |
| @return json | 回傳json格式 |
| @return mixed | 可能回傳不同的格式 |

2. 回傳為自己本身類別物件

|  |
| --- |
| /\*\*  \* Set which user the command should run as.  \*  \* @param string $user username  \* @return $this  \*/  public function user($user)  {  $this->user = $user;  return $this;  } |

3. 回傳為其他類別物件

**格式**：

|  |
| --- |
| @return ["Type"][<description>] |

**範例**：

1. 類別物件(class)

|  |
| --- |
| /\*\*  \* 取得訂單  \*  \* @return object class:Order 訂單 Model  \*/  private static function getLatestOrder()  {  return Order::orderBy('order\_date', 'desc')->limit(1)->get();  } |

1. 其他php內建函式物件

|  |
| --- |
| /\*\*  \* 取得XML解析後物件  \*  \* @param string $xmlstring XML字串  \* @return object SimpleXMLElement  \*/  public function convertStringToXML($xmlstring)  {  $xmlobject = simplexml\_load\_string($xmlstring);  return $xmlobject;  } |

4. 該方法處理無需回傳結果

|  |
| --- |
| /\*\*  \* 初始設置  \*  \* @return void  \*/  public static function init()  {  self::$word\_type = self::TYPE\_NUMBER;  self::$code = 'CaptchaCode' . date('Ymd');  } |

# 格式\_Formatting

### **透過 PHPSTORM** Code Style **設定**

File -> Settings -> Editor -> Code Style -> PHP

* 在 JetBrains 所開發的軟體中，自動排版程式碼
  + windows平台：Ctrl+Alt+L
  + mac平台：Command+Alt+L

### 透過PHP Code Sniffer 檢查規範

安裝方式請參考: <https://docs.google.com/document/d/1niajzVWZP899mffBGRKQAFuUqUDPkLgwpDrcTPcCDmg/edit#heading=h.8v881qj8qbyk>

# 參考

### PHP Standards Recommendations

<https://www.php-fig.org/psr/>