**CODING CONVENTION**

NỘI DUNG

[**CODING CONVENTION** 1](#_Toc419935529)

[**I.** **PHP File formatting** 2](#_Toc419935530)

[***1.*** ***Ký hiệu kết thúc file*** 2](#_Toc419935531)

[***3.*** ***Giới hạn code trên 1 dòng*** 2](#_Toc419935532)

[**II.** **NAMING CONVENTIONS** 2](#_Toc419935533)

[***1.*** ***Lớp*** 2](#_Toc419935534)

[***2.*** ***Interface*** 2](#_Toc419935535)

[***3.*** ***Tên thư mục, file*** 3](#_Toc419935536)

[***4.*** ***Hàm và phương thức*** 3](#_Toc419935537)

[***5.*** ***Biến*** 4](#_Toc419935538)

[***6.*** ***Hằng*** 6](#_Toc419935539)

[***7.*** ***My Sql Conventions*** 6](#_Toc419935540)

[**III. Coding style** 7](#_Toc419935541)

[***1.*** ***PHP Code Demarcation*** 7](#_Toc419935542)

[***2.*** ***String*** 7](#_Toc419935543)

[***3.*** ***Mảng*** 8](#_Toc419935544)

[***4.*** ***Lớp*** 8](#_Toc419935545)

[***5.*** ***Hàm và phương thức*** 9](#_Toc419935546)

[***6.*** ***Câu lệnh điều khiển*** 10](#_Toc419935547)

[***a)*** ***If / Else / Elseif*** 11](#_Toc419935548)

[***8.*** ***Cách trình bày cho các khối*** 13](#_Toc419935549)

[***9.*** ***Comment*** 14](#_Toc419935550)

1. **PHP File formatting**
2. ***Ký hiệu kết thúc file***

Đối với những file chỉ gồm code php không được dùng thẻ đóng (“?>”). Điều kiện trên không yêu cầu bởi PHP engine nhưng việc không dùng thẻ đóng tránh được việc bị tràn new line khi output ra HTML page

1. ***Quy định lề đầu dòng***

Sử dụng 1 tab cho mỗi level.

Trong các chuẩn coding khác thường quy ước dùng 4 khoảng trắng thay cho dung tab. Lý do của cách dung này là do trong các editor khác nhau độ dài của tab khác nhau dẫn đến có thể không thống nhất trong việc hiển thị code. Tuy nhiên do tính thống nhất trong môi trường lập trình tại trung tâm nên việc sử dụng Tab không bị các nhược điểm trên. Ngược lại có một số ưu điểm:

* Tăng tốc độ di chuyển của con trỏ thay vì phải di chuyển 4 space thì chỉ phải di chuyển 1 tab
* Tăng tốc độ thực hiện dòng code

1. ***Giới hạn code trên 1 dòng***

Việc giới hạn số ký tự trên 1 dòng code nhằm giúp cho lập trình viên nhìn code được dễ dàng. Một dòng code nên chỉ có 80 ký tự và tối đa là 120 ký tự.

1. **NAMING CONVENTIONS**
2. ***Lớp***

Chữ cái đầu tiên viết hoa, các từ được kết hợp bởi chữ cái la tinh viết liền nhau. Được phân biệt với nhau bàng chữ viết hoa.

Nếu tên lớp có từ viết tắt, thì viết hoa chữ đầu tiên của từ đó mà không được viết hoa toàn bộ từ.

Không được dùng dấu gạch dưới (‘\_’).

Ví dụ:

class GetHtmlStatistic       // NOT GetHTMLStatistic

class GetContent //Not Get\_Content

Lưu ý: Nếu hệ thống được chia thành nhiều namespace thì tên lớp phải bao gồm cả namespace, phân cách bởi dấu gạch dưới (‘\_’).

Ví dụ

class Xcms\_db\_DbManager // in xcms/db/db\_manager.php

class Xcms\_file\_XmlManager // in xcms/file/xml\_manager.php

* 1. ***Interface***

Quy tắc đặt tên tương tự class nhưng bắt đầu bởi ký tự ‘I’;

Ví dụ:

IXcms\_db\_Db

IXcms\_file\_Xml

* 1. ***Tên thư mục, file***

Tên thư mục và file viết chữ cái thường, các từ cách nhau bởi dấu gạch dưới (‘\_’).

Ví dụ:

xcms/db/db\_manager.php

xcms/file/xml\_manager.php

1. ***Hàm và phương thức***

Tên hàm và phương thức bắt đầu bằng động từ vì nó đại diện cho một hay một chuỗi hành động

**Phương thức (method)**

Chữ cái đầu tiên viết hoa, các từ được kết hợp bởi chữ cái la tinh viết liền nhau. Được phân biệt với nhau bằng chữ viết hoa.

Nếu tên phương thức có từ viết tắt, thì viết hoa chữ đầu tiên của từ đó mà không được viết hoa toàn bộ từ.

Không được dùng dấu gạch dưới (‘\_’).

Ví dụ:

class NameOneTwo

{

function DoIt() {};

function HandleError() {};

}

**Hàm (function)**

Tên hàm viết chữ thường, các từ cách nhau bởi dấu gạch dưới (‘\_’).

Ví dụ:

function some\_bloody\_function()

{

//some code here..

}

**Accessor method and function**

Accessor method & function có thể tam dịch là hàm (phương thức) truy suất để nói lên đặc tính của hàm hay phương thức đó dùng để định giá trị hay lấy giá trị trạng thái của hệ thống hay của lớp.

**Getters**

Các hàm lấy giá trị có tiếp đầu ngữ là ***get*** nếu theo sau nó là danh từ. Đối với dạng giá trị trả về là Boolean thì tiếp đầu ngữ là ***is*** nếu theo sau là tính từ, ***has*** nếu sau là danh từ và ***can*** nếu sau là động từ.

Ví dụ:

getFirstName()

getAccountNumber()

isPersistent()

hasDependents()

canPrint()

**Setters**

Các hàm (phương thức) setter là các hàm (phương thức) làm thay đổi một trường giá trị của hệ thống (lớp).

Ví dụ:

setFirstName(String aName)

setAccountNumber(int anAccountNumber)

setReasonableGoals(Vector newGoals)

* 1. ***Biến***

Với các biến thường được viết chữ thường và cách nhau bởi dấu (‘\_’).

Tên mảng đặt thêm tiếp đầu ngữ (‘arr\_’) để phân biệt với biến đơn.

Riêng các biến đặc biệt đều có tiếp đầu ngữ và viết hoa chữ cái đầu của mỗi từ, không dung dấu gạch dưới (‘\_’). Bao gồm:

**Thuộc tính (Class Attribute name)**

Bắt đầu bằng chữ cái ‘m’.

Ví dụ

class NameOneTwo

{

var $mVarAbc;

var $mErrorNumber;

var $mrName;

function VarAbc() {};

function ErrorNumber() {};

}

**Tham biến (reference variable)**

Bắt đầu bằng chữ cái ‘r’.

Ví dụ

class Test

{

var $mrStatus;

function DoSomething(&$rStatus) {};

function &rStatus() {};

}

**Biến toàn cục (global variable)**

Bắt đầu bằng chữ cái ‘g’.

Ví dụ

global $gLog;

global &$grLog;

**Biến tĩnh (static variable)**

Bắt đầu bằng chữ cái ‘s’.

Ví dụ

function test()  
{  
 static $msStatus = 0;

}

**Đối số của phương thức (Method Argument Names)**

Chữ cái đầu tiên là chữ thường, các từ tiếp theo viết hoa chữ cái đầu.

Không dùng dấu gạch dưới (‘\_’) .

Ví dụ

class NameOneTwo

{

function StartYourEngines(&$someEngine, &$anotherEngine) {

$this->mSomeEngine = $someEngine;

$this->mAnotherEngine = $anotherEngine;

}

…

}

* 1. ***Hằng***

Hằng số xử dụng chữ cái hoa và các từ cách nhau bởi dấu gạch dưới.

Ví dụ:

define("MAXSIZE", 100);

echo MAXSIZE;

class a {

const B = 'c';

}

echo constant('a::b');

* 1. ***My Sql Conventions***

**Tên bảng**

Tên bảng viết chữ thường, dùng danh từ số ít, các từ cách nhau bởi dấu cách (‘\_’)

Sử dụng tiếp đầu ngữ chung của module hoặc của dự án.

Đối với các bảng hệ thống (dung chung cho tất cả các dự án trong cùng Framework) dùng tiếp đầu ngữ là ‘sys\_’

Các bảng nối quan hệ giữa 2 bảng đặt tên sẽ gồm

‘<tiếp đầu ngữ>\_<tên bảng 1>\_<tên bảng 2>

Ví dụ

Table mod\_customer

Table mod\_rights

Table mod\_customer\_rights

Table sys\_configuration

**Tên trường**

Chữ cái thường cách nhau bởi dấu gạch dưới.

Tiếp đầu ngữ là tên bảng hoặc tên bảng rút gọn.

Các trường Foreign Key đặt tên theo đúng tên Primary Key ở bảng cha.

Ví dụ:

CREATE TABLE `mod\_session` (

`session\_id` int(100) NOT NULL auto\_increment,

`session\_in` datetime NOT NULL,

`session\_out` datetime default NULL,

`admin\_id` char(30) NOT NULL, //Foreign key

PRIMARY KEY  (`session\_id`)

)

**III. Coding style**

* 1. ***PHP Code Demarcation***

Khối code php được bao trong cặp thẻ <?php .. ?>. Không được dung thẻ rút gọn <? .. ?> cũng như <?=$var?> .

Đối với file chỉ gồm code PHP thì không dùng thẻ đóng ‘?>’, xem 2.1

* 1. ***String***

**String Literal**

String Literal là chuỗi không chứa các biến trong nó. String Literal nên đặt trong dấu nháy đơn để tránh việc chạy các đoạn script PHP.

Đặt một khoảng trắng trước và sau ký hiệu nối 2 string để dễ nhìn

Ví dụ:

$name = 'john';

$hello = 'hello $ name!';

echo $hello; //will out: hello $var

$hello = ‘hello’ . $name //will out: hello john

**Sql string**

Sql string là chuỗi dùng để truy vấn câu lệnh sql. Một số các quy tắc cho Sql string sau:

Viết hoa các Sql keywork.

Ngắt xuống dòng đối với các câu lệnh dài.

Ví dụ

// Short SQL Query

$sql = "SELECT name FROM people WHERE id = '" . $id . "'";

// Long SQL Query

$sql = “SELECT people\_id”

, people\_name

, people\_phone

FROM people

WHERE name = 'Fred'

OR name = 'Susan'

ORDER BY name ASC

LIMIT 20”;

Việc ngắt mỗi trường thành một dòng rất dễ nhìn đồng thời dễ cho việc sửa đổi câu lệnh, chỉ cần copy một dòng và sửa lại tên trường là có thể cho hiển thị thêm các trương khác trong câu truy vấn. Đồng thời việc để dấu phẩy (‘,’) đằng trước tên trường giúp cho việc copy thêm trường hiển thị vào cuối lệnh SELECT được thuận tiện hơn, không xảy ra lỗi.

Ví dụ:

// Long SQL Query

$sql = “SELECT people\_id, ”

. “ people\_name, ”

. “ people\_phone”

. “ people\_name, ” //copy from line 2 made error

. “FROM people ”;

* 1. ***Mảng***

Chỉ mục của mảng không được phép là số âm và default là từ 0.

Các giá trị của mảng cách nhau bởi dấu phẩy và nên tuân theo quy tắc văn bản: them một khoảng trắng sau dấu phẩy

Ví dụ:

$arr\_eg = array(1, 2, 3, 'Zend', 'Studio');

Nên khai báo mảng tường minh chỉ mục và giá trị theo định dạng sau:

$ arr\_eg = array('firstKey'  => 'firstValue'

, 'secondKey' => 'secondValue'

, 'thirdKey' => 'thirdValue'

, 'fourthKey' => 'fourthValue');

* 1. ***Lớp***

**Khai báo lớp**

Lớp được đặt tên theo đúng [naming conventions.](https://docs.google.com/document/d/1OyIqKGVB6nElDPVH9gpk2KFnGNL117q6EsVTs0wjzeY/edit#bookmark=id.d14450439c11)

Trước khai báo class luôn đặt một khối comment (xem cụ thể 4.7).

Mỗi file chỉ nên chứa 1 class và không chứa các dòng code ngoài class.

/\*\*

\* Documentation Block Here

\*/

class SampleClass

{

// entire content of class

// must be indented four spaces

}

**Biến**

Tên biến trong class được đặt theo đúng [naming conventions.](https://docs.google.com/document/d/1OyIqKGVB6nElDPVH9gpk2KFnGNL117q6EsVTs0wjzeY/edit#bookmark=id.1b871b1b1b31)

Biến phải được khai báo trước mọi phương thức.

Nên có khối comment trước mỗi biến quan trọng.

/\*\*

\* Class Docblock Here

\*/

class Test

{

/\*\*

\* Variable Docblock Here

\*/

var $mAbc = null;

/\*\*

\* Variable Docblock Here

\*/

var $mCdf = null;

}

* 1. ***Hàm và phương thức***

Tên hàm và phương thức đặt theo đúng [naming conventions.](https://docs.google.com/document/d/1OyIqKGVB6nElDPVH9gpk2KFnGNL117q6EsVTs0wjzeY/edit#bookmark=id.8068842b3c5a)

Trước mỗi khai báo hàm (phương thức) đặt một khối comment về hàm (phương thức) đó.

Các function cách nhau một dòng trắng.

/\*\*

\* Class Docblock Here

\*/

class Buxa\_Foo

{

/\*\*

\* Method Docblock Here

\*/

function SampleMethod($a)

{

// Entire content of function must be indented four spaces

}

/\*\*

\* Method Docblock Here

\*/

function AnotherMethod()

{

// ...

}

/\*\*

\* Method Docblock Here

\*/

function AnotherNewMethod(Array $b)

{

// ...

}

}

* 1. ***Câu lệnh điều khiển***
     1. ***If / Else / Elseif***

Khối điều kiện đặt trong cặp ngoặc tròn “()” và có một khoảng trắng trước từ khóa if để dễ nhìn.

Cặp {} được trình bày như ví dụ dưới:

if (condition)                 // Comment

{

}

else if (condition)          // Comment

{

}

else                             // Comment

{

}

Nếu bạn sử dụng câu lệnh “if … then” thì là 1 ý kiến hay và bạn luôn luôn phải có 1 block khác cho việc tìm ra những trường hợp không thực hiện được. Hãy đưa vào 1 thông báo khi có hành động không chính xác sảy ra.

* + 1. **Switch**

Khối điều kiện đặt trong cặp ngoặc tròn “()” và có một khoảng trắng trước từ khóa “switch”.

Tất cả nội dung bên trong mệnh đề “switch” đều phải thụt vào 1 tab. Nội dung bên dưới mỗi mệnh đề  “case” đều phải thụt vào thêm 1 tab nữa

Cấu trúc “default” không được phép bỏ qua trong câu lệnh “switch”.

Chú ý: đôi khi sẽ rất hữu ích khi ta viết câu lệnh “case” trong những trường hợp tiếp sau không gồm “break” hoặc “return”. Để phân biệt  những trường hợp này với các lỗi, mỗi mệnh đề “case” phải bao gồm cả những dòng chú thích

Ví dụ

switch ($sampleNumber)

{  
    case 1:  
        // ...  
        break;  
  
    case 2:  
        // ...  
        break;  
  
    default:  
        // ...  
        break;  
}

* + 1. **While, do while**

-1 câu lệnh while giống như sau (chú ý vị trí của các dấu ngoặc và khoảng trống):

<?php

while ($expr) {

// structure body

}

-Tương tự, 1 câu lệnh do while giống như sau (chú ý vị trí của các dấu ngoặc và khoảng trống):

<?php

do {

// structure body;

} while ($expr);

* + 1. **Foreach**

-1 câu lệnh foreach giống như sau (chú ý vị trí của các dấu ngoặc và khoảng trống):

<?php

foreach ($iterable as $key => $value) {

// foreach body

}

* + 1. **Try, catch**

-1 khối try, catch giống như sau (chú ý vị trí của các dấu ngoặc và khoảng trống):

<?php

try {

// try body

} catch (FirstExceptionType $e) {

// catch body

} catch (OtherExceptionType $e) {

// catch body

}

1. **Continue,break and ?:**

**Continue và Break**

Hạn chế tối đa việc dùng continue và break. Thường các khối lệnh dùng continue và break đều có thể khử bằng việc sử dụng các câu lệnh lặp và rẽ nhánh.

Nếu sử dụng continue và break phải comment về flow.

**?:**

Vấn đề là mọi thường cố gắng và nhồi nhét quá nhiều đoạn code giữa dấu ? và dấu :. Sau đây là 1 số quy tắc:

* Đặt điều kiện chính nổi bật hơn để thiết lập những đoạn code khác
* Nếu có thể, các actions cho việc kiểm tra nên xây dựng theo function đơn giản.
* Đặt những action nay trong câu lệnh "Then" và  "Else" trong các dòng tách biệt nếu không thì có thể đặt chúng rõ ràng trong cùng 1 dòng.

Ví dụ

(condition) ? funct1() : func2();

hoặc đặt như sau nếu quá dài

(condition)

? long statement

: another long statement;

* 1. ***Cách trình bày cho các khối***
* Các khối nên được trình bày cho đúng canh lề level.
* Rõ ràng.
* Tương tự các block khởi tạo của biến cũng nên được xếp thành bảng.
* Dấu hiệu ‘&’ nên được sử dụng liền sát với kiểu , không liền sát với tên.

Ví dụ

var        $mDate

var&      $mrDate

var&      $mrName

var        $mName

$mDate      = 0;

$mrDate     = NULL;

$mrName   = 0;

$mName     = NULL;

Mỗi phát biểu trên 1 hàng

Sau tên biến là dấu tab(4 khoảng trắng – vì nhiều biến có tên dài) sau đó tới dấu = và tiếp theo là 1 khoảng trắng rồi đến gtrị.

Nên để duy nhất 1 phát biểu trên 1 hàng trừ khi những phát biểu đó có liên quan chặt chẽ với nhau.

* 1. ***Comment***
     + 1. **Documentation Format**

Tất cả các khối chú thích (“docblock”) phải tương thích định dạng phpDocumentor. Đặc tả chuẩn chú thích của phpDocument không nằm trong phạm vi của tài liệu này, các thông tin chi tiết có thể tìm hiểu tại <http://phpdoc.org>. Sau đây là một số chuẩn quy định trong tài liệu này:

* + - 1. **Files**

Tất cả các file chứa mã PHP đều phải có khối chú thích đầu file, tối thiểu gồm các trường thông tin sau:

/\*\*  
 \* Short description for file  
 \*  
 \* Long description for file (if any)...  
 \*  
 \* LICENSE: Some license information  
 \*  
 \* @copyright  2006 XCMS  
 \* @license    XCMS Licence 1.0  
 \* @version    1.0  
 \* @link       http://qsoftvietnam.com/products.php?id=c81e728d9d4c2f636f067f89cc14862c  
 \* @since      File available since Release 1.0  
 \*/

* + - 1. **Lớp**

Tất cả các lớp chứa mã PHP đều phải có khối chú thích đầu class, tối thiểu gồm các trường thông tin sau:

/\*\*  
 \* Short description for class  
 \*  
 \* Long description for class (if any)...  
 \*  
 \* @copyright  2006 XCMS  
 \* @license    XCMS Licence 1.0  
 \* @version    Release: @1.0@  
 \* @link       http://qsoftvietnam.com/products.php?id=c81e728d9d4c2f636f067f89cc14862c  
 \* @since      Class available since Release 1.0  
 \*/

* + - 1. **Hàm**

Tất cả các hàm bao gồm cả phương thức của lớp đều phải có chú thích gồm tối thiểu các trường sau: Description, param(nếu có), Return Values.

/\*\*  
 \* Short description for the function  
 \*  
 \* Long description for the function (if any)...  
 \*  
 \* @param  array  $array  Description of array  
 \* @param  string $string Description of string  
 \* @return boolean  
 \*/

Nếu hàm throw an exeption phải sử dụng thêm trường “@throws”

@throws Exception\_Class\_Name Description

* + - 1. **Biến**

Các biến thuộc class phải có khối chú thích miêu tả và kiểu biến

/\*\*  
 \* Description for the variable  
 \* @var array  
 \*/

* + - 1. **Comment Keywords**
* **:TODO: topic**  
  Có rất nhiều vấn đề ở đây.
* **:BUG: [bugid] topic**  
  Vấn đề về các lỗi, giải thích chúng.
* **:KLUDGE:**  
  Khi bạn làm một việc gì đó rất tệ và bạn cần giải thích bạn làm cách nào để nó tốt hơn nếu bạn có thêm 1 chút thời.
* **:TRICKY:**  
  Nói cho mọi người biết đoạn code sau đây rất tồi, vì vậy đừng sửa chữa nó nếu không suy nghĩ.
* **:WARNING:**  
  Thận trọng với 1 cái gì đó.
* **:PARSER:**  
  Đôi khi bạn cần phải phân tích lại vấn đề, hãy lấy dẫn chứng bằng tài liệu, vấn đề sẽ được giải quyết thấu đáo.
* **:ATTRIBUTE: value**  
  Theo cách thông thường của một thuộc tính nhúng trong chú thích. Bạn có thể tự tạo ra các thuộc tính và chúng sẽ rất thú vị.

Ví dụ

// :TODO: tmh 960810: possible performance problem

// We should really use a hash table here but for now we'll

// use a linear search.

// :KLUDGE: tmh 960810: possible unsafe type cast

// We need a cast here to recover the derived type. It should

// probably use a virtual method or template.