# 基本代码规范

本节我们将会讨论一些基本的代码规范问题，以此作为将来讨论更高级别的代码分享和技术互用的基础。

[RFC 2119](http://www.ietf.org/rfc/rfc2119.txt)中的必须(MUST)，不可(MUST NOT)，建议(SHOULD)，不建议(SHOULD NOT)，可以/可能(MAY)等关键词将在本节用来做一些解释性的描述。

## 1. 概述

* 源文件必须只使用 <?php 和 <?= 这两种标签。
* 源文件中php代码的编码格式必须只使用不带字节顺序标记(BOM)的UTF-8。
* 一个源文件建议只用来做声明（类(class)，函数(function)，常量(constant)等）或者只用来做一些引起副作用的操作（例如：输出信息，修改.ini配置等）,但不建议同时做这两件事。
* 命名空间(namespace)和类(class) 必须遵守[PSR-0](https://github.com/hfcorriez/fig-standards/blob/zh_CN/%E6%8E%A5%E5%8F%97/PSR-0.md)标准。
* 类名(class name) 必须使用骆驼式(StudlyCaps)写法 (译者注：驼峰式(cameCase)的一种变种，后文将直接用StudlyCaps表示)。
* 类(class)中的常量必须只由大写字母和下划线(\_)组成。
* 方法名(method name) 必须使用驼峰式(cameCase)写法(译者注：后文将直接用camelCase表示)。

## 2. 文件

### 2.1. PHP标签

PHP代码必须只使用长标签(<?php ?>)或者短输出式标签(<?= ?>)；而不可使用其他标签。

### 2.2. 字符编码

PHP代码的编码格式必须只使用不带字节顺序标记(BOM)的UTF-8。

### 2.3. 副作用

一个源文件建议只用来做声明（类(class)，函数(function)，常量(constant)等）或者只用来做一些引起副作用的操作（例如：输出信息，修改.ini配置等）,但不建议同时做这两件事。

短语副作用(side effects)的意思是 在包含文件时 所执行的逻辑与所声明的类(class)，函数(function)，常量(constant)等没有直接的关系。

副作用(side effects)包含但不局限于：产生输出，显式地使用require或include，连接外部服务，修改ini配置，触发错误或异常，修改全局或者静态变量，读取或修改文件等等

下面是一个既包含声明又有副作用的示例文件；即应避免的例子：

<?php

// 副作用：修改了ini配置

ini\_set('error\_reporting', E\_ALL);

// 副作用：载入了文件

include "file.php";

// 副作用：产生了输出

echo "<html>\n";

// 声明

function foo()

{

// 函数体

}

下面是一个仅包含声明的示例文件；即应提倡的例子：

<?php

// 声明

function foo()

{

// 函数体

}

// 条件式声明不算做是副作用

if (! function\_exists('bar')) {

function bar()

{

// 函数体

}

}

## 3. 空间名(namespace)和类名(class name)

命名空间(namespace)和类(class)必须遵守 [PSR-0](https://github.com/hfcorriez/fig-standards/blob/zh_CN/%E6%8E%A5%E5%8F%97/PSR-0.md).

这意味着一个源文件中只能有一个类(class)，并且每个类(class)至少要有一级空间名（namespace）：即一个顶级的组织名(vendor name)。

类名(class name) 必须使用StudlyCaps写法。

PHP5.3之后的代码必须使用正式的命名空间(namespace) 例子：

<?php

// PHP 5.3 及之后:

namespace Vendor\Model;

class Foo

{

}

PHP5.2.x之前的代码建议用伪命名空间Vendor\_作为类名(class name)的前缀

<?php

// PHP 5.2.x 及之前:

class Vendor\_Model\_Foo

{

}

## 4. 类的常量、属性和方法

术语类(class)指所有的类(class)，接口(interface)和特性(trait)

### 4.1. 常量

类常量必须只由大写字母和下划线(\_)组成。 例子：

<?php

namespace Vendor\Model;

class Foo

{

const VERSION = '1.0';

const DATE\_APPROVED = '2012-06-01';

}

### 4.2. 属性

本指南中故意不对$StulyCaps，$camelCase或者$unser\_score中的某一种风格作特别推荐，完全由读者依据个人喜好决定属性名的命名风格。

但是不管你如何定义属性名，建议在一个合理的范围内保持一致。这个范围可能是组织(vendor)级别的，包(package)级别的，类(class)级别的，或者方法(method)级别的。

### 4.3. 方法

方法名则必须使用camelCase()风格来声明。

**基本代码规范**

本篇规范制定了代码基本元素的相关标准，  
以确保共享的PHP代码间具有较高程度的技术互通性。

关键词 “必须”("MUST")、“一定不可/一定不能”("MUST NOT")、“需要”("REQUIRED")、  
“将会”("SHALL")、“不会”("SHALL NOT")、“应该”("SHOULD")、“不该”("SHOULD NOT")、  
“推荐”("RECOMMENDED")、“可以”("MAY")和”可选“("OPTIONAL")的详细描述可参见 [RFC 2119](http://www.ietf.org/rfc/rfc2119.txt) 。

**1. 概览**

* PHP代码文件**必须**以 <?php 或 <?= 标签开始；
* PHP代码文件**必须**以 不带BOM的 UTF-8 编码；
* PHP代码中**应该**只定义类、函数、常量等声明，或其他会产生 从属效应 的操作（如：生成文件输出以及修改.ini配置文件等），二者只能选其一；
* 命名空间以及类**必须**符合 PSR 的自动加载规范：[PSR-0](https://github.com/PizzaLiu/PHP-FIG/blob/master/PSR-0-cn.md" \t "_blank) 或 [PSR-4](http://segmentfault.com/blog/gtg/1190000002521658) 中的一个；
* 类的命名**必须**遵循 StudlyCaps 大写开头的驼峰命名规范；
* 类中的常量所有字母都**必须**大写，单词间用下划线分隔；
* 方法名称**必须**符合 camelCase 式的小写开头驼峰命名规范。

**2. 文件**

**2.1. PHP标签**

PHP代码**必须**使用 <?php ?> 长标签 或 <?= ?> 短输出标签；  
**一定不可**使用其它自定义标签。

**2.2. 字符编码**

PHP代码**必须**且只可使用不带BOM的UTF-8编码。

**2.3. 从属效应（副作用）**

一份PHP文件中**应该**要不就只定义新的声明，如类、函数或常量等不产生从属效应的操作，要不就只有会产生从属效应的逻辑操作，但**不该**同时具有两者。

“从属效应”(side effects)一词的意思是，仅仅通过包含文件，不直接声明类、  
函数和常量等，而执行的逻辑操作。

“从属效应”包含却不仅限于：生成输出、直接的 require 或 include、连接外部服务、修改 ini 配置、抛出错误或异常、修改全局或静态变量、读或写文件等。

以下是一个反例，一份包含声明以及产生从属效应的代码：

<?php

// 从属效应：修改 ini 配置

ini\_set('error\_reporting', E\_ALL);

// 从属效应：引入文件

include "file.php";

// 从属效应：生成输出

echo "<html>\n";

// 声明函数

function foo()

{

// 函数主体部分

}

下面是一个范例，一份只包含声明不产生从属效应的代码：

<?php

// 声明函数

function foo()

{

// 函数主体部分

}

// 条件声明\*\*不\*\*属于从属效应

if (! function\_exists('bar')) {

function bar()

{

// 函数主体部分

}

}

**3. 命名空间和类**

命名空间以及类的命名必须遵循 [PSR-0](https://github.com/PizzaLiu/PHP-FIG/blob/master/PSR-0-cn.md).

根据规范，每个类都独立为一个文件，且命名空间至少有一个层次：顶级的组织名称（vendor name）。

类的命名必须 遵循 StudlyCaps 大写开头的驼峰命名规范。

PHP 5.3及以后版本的代码**必须**使用正式的命名空间。

例如：

<?php

// PHP 5.3及以后版本的写法

namespace Vendor\Model;

class Foo

{

}

5.2.x及之前的版本**应该**使用伪命名空间的写法，约定俗成使用顶级的组织名称（vendor name）如 Vendor\_ 为类前缀。

<?php

// 5.2.x及之前版本的写法

class Vendor\_Model\_Foo

{

}

**4. 类的常量、属性和方法**

此处的“类”指代所有的类、接口以及可复用代码块（traits）

**4.1. 常量**

类的常量中所有字母都**必须**大写，词间以下划线分隔。  
参照以下代码：

<?php

namespace Vendor\Model;

class Foo

{

const VERSION = '1.0';

const DATE\_APPROVED = '2012-06-01';

}

**4.2. 属性**

类的属性命名可以遵循 大写开头的驼峰式 ($StudlyCaps)、小写开头的驼峰式 ($camelCase) 又或者是 下划线分隔式 ($under\_score)，本规范不做强制要求，但无论遵循哪种命名方式，都**应该**在一定的范围内保持一致。这个范围可以是整个团队、整个包、整个类或整个方法。

**4.3. 方法**

方法名称**必须**符合 camelCase() 式的小写开头驼峰命名规范。