**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。

**关于「能愿动词」的使用**

为了避免歧义，文档大量使用了「能愿动词」，对应的解释如下：

* 必须 (MUST)：绝对，严格遵循，请照做，无条件遵守；
* 一定不可 (MUST NOT)：禁令，严令禁止；
* 应该 (SHOULD) ：强烈建议这样做，但是不强求；
* 不该 (SHOULD NOT)：强烈不建议这样做，但是不强求；
* 可以 (MAY) 和 可选 (OPTIONAL) ：选择性高一点，在这个文档内，此词语使用较少；

**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 副作用 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-4]() 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. 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 副作用 的操作，要不就只书写会产生 副作用 的逻辑操作，但 **不该** 同时具有两者。

「副作用」(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。

「副作用」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「函数声明」以及产生「副作用」的代码：

<?php

// 「副作用」：修改 ini 配置

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

// 「副作用」：引入文件

include "file.php";

// 「副作用」：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「副作用」的代码：

<?php

// 声明函数

function foo()

{

// 函数主体部分

}

// 条件声明 \*\*不\*\* 属于「副作用」

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-4]()。

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)
* 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。