X2 Force Coding Standards

1. Overview

- Files MUST use only <?php and <?= tags.
- Files SHOULD *either* declare symbols (classes, functions, constants, etc.) *or* cause side-effects (e.g. generate output, etc.) but SHOULD NOT do both.
- Namespaces and classes MUST follow an "autoloading"
- Class names MUST be declared in StudlyCaps.
- Class constants MUST be declared in all uppercase with underscore separators.
- Method names MUST be declared in camelCase.

2. Files

2.1. PHP Tags

PHP code MUST use the long <?php ?> tags or the short-echo <?= ?> tags; it MUST NOT use the other tag variations.

2.2. Side Effects

A file SHOULD declare new symbols (classes, functions, constants, etc.) and cause no other side effects, or it SHOULD execute logic with side effects, but SHOULD NOT do both.

The phrase "side effects" means execution of logic not directly related to declaring classes, functions, constants, etc., *merely from including the file*.

"Side effects" include but are not limited to: generating output, explicit use of require or include, connecting to external services, modifying ini settings, emitting errors or exceptions, modifying global or static variables, reading from or writing to a file, and so on.

3. Namespace and Class Names

Namespaces and classes MUST follow an "autoloading"

This means each class is in a file by itself, and is in a namespace of at least one level: a top-level vendor name.

Class names MUST be declared in StudlyCaps.

Code written for PHP 5.3 and after MUST use formal namespaces.

For example:

```
<?php
// PHP 5.3 and later:
namespace Vendor\Model;

class Foo {
}
```

Code written for 5.2.x and before SHOULD use the pseudo-namespacing convention of Vendor_ prefixes on class names.

```
<?php
// PHP 5.2.x and earlier:
class Vendor_Model_Foo {
}
```

4. Class Constants, Properties, and Methods

The term "class" refers to all classes, interfaces, and traits.

4.1. Constants

Class constants MUST be declared in all uppercase with underscore separators. For example:

```php
<?php
namespace Vendor\Model;

class Foo {
    const VERSION = '1.0';
    const DATE_APPROVED = '2012-06-01';
}
```

4.2. Properties

This guide intentionally avoids any recommendation regarding the use of $StudlyCaps, $camelCase, or $under_score property names.

Whatever naming convention is used SHOULD be applied consistently within a reasonable scope. That scope may be vendor-level, package-level, class-level, or method-level.

4.3. Methods

Method names MUST be declared in camelCase().

5. Formatting

Code MUST use 4 spaces for indenting, not tabs.

Lines SHOULD be 80 characters or less.

There MUST be one blank line after the namespace declaration, and there MUST be one blank line after the block of use declarations.

Opening braces for classes MUST go on the next line.

Closing braces for classes MUST go on the next line after the body.

Opening braces for methods MUST go on the next line,

Closing braces for methods MUST go on the next line after the body.

Visibility MUST be declared on all properties and methods; abstract and final MUST be declared before the visibility; static MUST be declared after the visibility.

Control structure keywords MUST have one space after them; method and function calls MUST NOT.

Opening braces for control structures MUST go on the same line.

Closing braces for control structures MUST go on the next line after the body.

Opening parentheses for control structures MUST NOT have a space after them.

Closing parentheses for control structures MUST NOT have a space before them.

Works Cited

We used the following sources to help make our Coding Standards:

1) PHP-Fig
2) GitHub