Cadê Meu HOSPITAL?

Alunos: Átilla Gallio, Fagner Rodrigues, Geison de Souza e Matheus Mello
UnB
–
FGA, Técnicas de Programação 1º/2014

Style Guide Document

1. Files

1.1. PHP Tags

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

PHP keywords MUST be in lower case.

The PHP constants `true`, `false`, and `null` MUST be in lower case.

1.2. Character Encoding

PHP code MUST use only UTF-8 without BOM.

1.3. Lines

All PHP files MUST end with a single blank line.

Lines SHOULD NOT be longer than 80 characters; lines longer than that SHOULD be split into multiple subsequent lines of no more than 80 characters each.

There MUST NOT be more than one statement per line.

2. Namespace and Class Names

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`.

For example:

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

class Foo
{
}
```

3. Class Constants, Properties, and Methods

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

3.1. Constants

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

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

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

3.2. Methods

Method names MUST be declared in `camelCase()`.

Method names MUST NOT be declared with a space after the method name. The opening brace MUST go on its own line, and the closing brace MUST go on the next line following the body. There MUST NOT be a space after the opening parenthesis, and there MUST NOT be a space before the closing parenthesis.

A method declaration looks like the following. Note the placement of parentheses, commas, spaces, and braces:

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

class ClassName
{
    public function fooBarBaz($arg1, &$arg2, $arg3 = [])
    {
        // method body
    }
}
```

In the argument list, there MUST NOT be a space before each comma, and there MUST be one space after each comma.

Method arguments with default values MUST go at the end of the argument list.

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

class ClassName
{
    public function foo($arg1, &$arg2, $arg3 = [])
    {
        // method body
    }
}
```

3.3. `abstract`, `final`, and `static`

When present, the `abstract` and `final` declarations MUST precede the visibility declaration.

When present, the `static` declaration MUST come after the visibility declaration.

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

abstract class ClassName
{
    protected static $foo;

    abstract protected function zim();

    final public static function bar()
    {
        // method body
    }
}
```

3.4. Extends and Implements

The `extends` and `implements` keywords MUST be declared on the same line as the class name.

The opening brace for the class MUST go on its own line; the closing brace for the class MUST go on the next line after the body.

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

use FooClass;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements \ArrayAccess, \Countable
{
    // constants, properties, methods
}
```

Lists of `implements` MAY be split across multiple lines, where each subsequent line is indented once. When doing so, the first item in the list MUST be on the next line, and there MUST be only one interface per line.

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

use FooClass;
use BarClass as Bar;
use OtherVendor\OtherPackage\BazClass;

class ClassName extends ParentClass implements
    \ArrayAccess,
    \Countable,
    \Serializable
{
    // constants, properties, methods
}
```

4. Control Structures

- There MUST be one space after the control structure keyword
- There MUST NOT be a space after the opening parenthesis
- There MUST NOT be a space before the closing parenthesis
- There MUST be one space between the closing parenthesis and the opening brace
- The structure body MUST be indented once
- The closing brace MUST be on the next line after the body

The body of each structure MUST be enclosed by braces. This standardizes how the structures look, and reduces the likelihood of introducing errors as new lines get added to the body.

4.1. `if`, `elseif`, `else`

An `if` structure looks like the following. Note the placement of parentheses, spaces, and braces; and that `else` and `elseif` are on the same line as the closing brace from the earlier body.

```php
<?php
if ($expr1) {
    // if body
} elseif ($expr2) {
    // elseif body
} else {
    // else body;
}
```

4.2. `switch`, `case`

A `switch` structure looks like the following. Note the placement of parentheses, spaces, and braces. The `case` statement MUST be indented once from `switch`, and the `break` keyword (or other terminating keyword) MUST be indented at the same level as the `case` body. There MUST be a comment such as `// no break` when fall-through is intentional in a non-empty `case` body.

```php
<?php
switch ($expr) {
    case 0:
        echo 'First case, with a break';
        break;
    case 1:
        echo 'Second case, which falls through';
        // no break
    case 2:
    case 3:
    case 4:
        echo 'Third case, return instead of break';
        return;
    default:
        echo 'Default case';
        break;
}
```

4.3. `while`, `do while`

A `while` statement looks like the following. Note the placement of parentheses, spaces, and braces.

```php
<?php
while ($expr) {
    // structure body
}
```

```php
<?php
do {
    // structure body;
} while ($expr);
```

4.4. `for`

A `for` statement looks like the following. Note the placement of parentheses, spaces, and braces.

```php
<?php
for ($i = 0; $i < 10; $i++) {
    // for body
}
```

4.5. `foreach`

A `foreach` statement looks like the following. Note the placement of parentheses, spaces, and braces.

```php
<?php
foreach ($iterable as $key => $value) {
    // foreach body
}
```

4.6. `try`, `catch`

A `try catch` block looks like the following. Note the placement of parentheses, spaces, and braces.

```php
<?php
try {
    // try body
} catch (FirstExceptionType $e) {
    // catch body
} catch (OtherExceptionType $e) {
    // catch body
}
```

5. Comments

5.1. `Headers`

Should follow this example comment.

```
/*  File: photos.php
    Description: This file contains functions related to
    uploading and displaying photos.
*/
```

5.2. `Functions`

Should follow this example comment.

```
// Upload a photo to the server so that you can later <display> it.

public function upload($file_name, $new_name, $new_width, new_$height, $directory)
    {
        ...
        ...
        returns true or false.
    }
```