Estándar de Programación

Lenguaje PHP

Sistema VASPA

VASPA Team

Nicolás Sartini



Este documento refleja los estándares de codificación del lenguaje PHP presentados en Estándar de Programación Coppel para PHP.

El mismo no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también hace referencia al orden y legibilidad del código escrito.



Tabla de contenido

[Introducción 5](#_Toc18342403)

[General 5](#_Toc18342404)

[Etiquetas PHP 5](#_Toc18342405)

[Codificación de Caracteres 5](#_Toc18342406)

[Archivos 5](#_Toc18342407)

[Líneas 5](#_Toc18342408)

[Indentación 6](#_Toc18342409)

[Palabras clave y true, false, null 6](#_Toc18342410)

[Espacio de nombre y declaraciones use 6](#_Toc18342411)

[Clases, Propiedades y Métodos 7](#_Toc18342412)

[Extensiones e implementaciones 7](#_Toc18342413)

[Propiedades 8](#_Toc18342414)

[Métodos 9](#_Toc18342415)

[Argumentos de los Métodos 9](#_Toc18342416)

[Abstract, final y static 11](#_Toc18342417)

[Llamadas a métodos y funciones 11](#_Toc18342418)

[Estructuras de Control 12](#_Toc18342419)

[Sentencias If, elseif, else 13](#_Toc18342420)

[Sentencias Switch, case 13](#_Toc18342421)

[Sentencias While, do while 14](#_Toc18342422)

[Sentencias For 15](#_Toc18342423)

[Sentencias Foreach 15](#_Toc18342424)

[Sentencias Try, catch 15](#_Toc18342425)

[Closures 16](#_Toc18342426)

[Comentarios 19](#_Toc18342427)

[Formatos de los tipos de comentarios 19](#_Toc18342428)

[Comentarios de una sola línea 20](#_Toc18342429)

[Comentarios multilineas 20](#_Toc18342430)

[Convenciones del código PHP 20](#_Toc18342431)

[Referencias: 20](#_Toc18342432)

Estándar de Programación

Lenguaje PHP

Introducción

El presente documento detalla las pautas generales a ser consideradas en la codificación, como se mencionan en el Estándar de Programación Coppel para PHP.

General

Etiquetas PHP

El código PHP debe utilizar las etiquetas largas <?php ?> o las etiquetas cortas para imprimir salida de información <?= ?>; No debe emplear otras variantes.

Codificación de Caracteres

El código PHP debe utilizar codificación UTF-8 sin BOM.

Archivos

* Todos los archivos PHP deben usar el final de línea Unix LF.
* Todos los archivos PHP deben terminar con una línea en blanco.
* La etiqueta de cierre ?> debe ser omitida en los archivos que sólo contengan código PHP.

Líneas

* No debe haber un límite estricto en la longitud de la línea.
* El límite flexible de la línea debe estar en 120 caracteres.
* Las líneas no deberían ser más largas de 80 caracteres; las líneas más largas de 80 caracteres deberían dividirse en múltiples líneas de no más de 80 caracteres cada una.
* No debe haber espacios en blanco al final de las líneas que no estén vacías.
* Pueden añadirse líneas en blanco para mejorar la lectura del código y para indicar bloques de código que estén relacionados.
* No debe haber más de una sentencia por línea.

Indentación

El código debe usar una indentación de cuatro espacios, y no debe usar tabuladores para la indentación.

Palabras clave y true, false, null

Las palabras clave de PHP deben estar en minúsculas.

Las constantes de PHP true, false y null deben estas en minúsculas.

Espacio de nombre y declaraciones use

* Cuando esté presente, debe haber una línea en blanco después de la declaración del namespace.
* Cuando estén presentes, todas las declaraciones use deben ir después de la declaración del namespace.
* Debe haber un use por declaración.
* Debe haber una línea en blanco después del bloque de declaraciones use.

Por ejemplo:

**<?php**

**namespace** Proveedor\Paquete;

**use** FooClass;

**use** BarClase **as** Bar;

**use** OtroProveedor\OtroPaquete\BazClase;

*// ... código PHP adicional ...*

Clases, Propiedades y Métodos

El término "clase" hace referencia a todas las clases, interfaces o traits.

Extensiones e implementaciones

Las palabras clave extends e implements deben declararse en la misma línea del nombre de la clase.

La llave de apertura de la clase debe ir en la línea siguiente; la llave de cierre debe ir en la línea siguiente al cuerpo de la clase.

**<?php**

**namespace** Proveedor\Paquete;

**use** FooClase;

**use** BarClase **as** Bar;

**use** OtroProveedor\OtroPaquete\BazClase;

**class** **NombreDeClase** **extends** ClasePadre **implements** \ArrayAccess, \Countable

{

*// constantes, propiedades, métodos*

}

La lista de implements puede ser dividida en múltiples líneas, donde las líneas subsiguientes serán indentadas una vez. Al hacerlo, el primer elemento de la lista debe estar en la línea siguiente, y debe haber una sola interfaz por línea.

**<?php**

**namespace** Proveedor\Paquete;

**use** FooClase;

**use** BarClase **as** Bar;

**use** OtroProveedor\OtroPaquete\BazClase;

**class** **NombreDeClase** **extends** ClasePadre **implements**

\ArrayAccess,

\Countable,

\Serializable

{

*// constantes, propiedades, métodos*

}

Propiedades

* La visibilidad debe ser declarada en todas las propiedades.
* La palabra clave var no debe ser usada para declarar una propiedad.
* No debe declararse más de una propiedad por sentencia.
* Los nombres de las propiedades no deberían usar un guión bajo como prefijo para indicar si son privadas o protegidas.

Una declaración de propiedades tendrá el siguiente aspecto:

**<?php**

**namespace** Proveedor\Paquete;

**class** **NombreDeClase**

{

**public** $foo **=** **null**;

}

Métodos

* La visibilidad debe ser declarada en todos los métodos.
* Los nombres de los métodos no deberían usar un guión bajo como prefijo para indicar si son privados o protegidos.
* Los nombres de métodos no deben estar declarados con un espacio después del nombre del método. La llave de apertura debe situarse en su propia línea, y la llave de cierre debe ir en la línea siguiente al cuerpo del método. No debe haber ningún espacio después del paréntesis de apertura, y no debe haber ningún espacio antes del paréntesis de cierre.
* La declaración de un método tendrá el siguiente aspecto. Fíjese en la situación de los paréntesis, las comas, los espacios y las llaves:

**<?php**

**namespace** Proveedor\Paquete;

**class** **NombreDeClase**

{

**public** **function** **fooBarBaz**($arg1, **&**$arg2, $arg3 **=** [])

{

*// cuerpo del método*

}

}

Argumentos de los Métodos

En la lista de argumentos no debe haber un espacio antes de cada coma y debe haber un espacio después de cada coma.

Los argumentos con valores por defecto del método deben ir al final de la lista de argumentos.

**<?php**

**namespace** Proveedor\Paquete;

**class** **NombreDeClase**

{

**public** **function** **foo**($arg1, **&**$arg2, $arg3 **=** [])

{

*// cuerpo del método*

}

}

La lista de argumentos puede dividirse en múltiples líneas, donde cada línea será indentada una vez. Cuando se dividan de esta forma, el primer argumento debe estar en la línea siguiente, y debe haber sólo un argumento por línea.

Cuando la lista de argumentos se divide en varias líneas, el paréntesis de cierre y la llave de apertura deben estar juntos en su propia línea separados por un espacio.

**<?php**

**namespace** Proveedor\Paquete;

**class** **NombreDeClase**

{

**public** **function** **metodoConNombreLargo**(

ClassTypeHint $arg1,

**&**$arg2,

**array** $arg3 **=** []

) {

*// cuerpo del método*

}

}

Abstract, final y static

Cuando estén presentes las declaraciones abstract y final, deben preceder a la declaración de visibilidad.

Cuando esté presente la declaración static, debe ir después de la declaración de visibilidad.

**<?php**

**namespace** Proveedor\Paquete;

**abstract** **class** **NombreDeClase**

{

**protected** **static** $foo;

**abstract** **protected** **function** **zim**();

**final** **public** **static** **function** **bar**()

{

*// cuerpo del método*

}

}

Llamadas a métodos y funciones

Cuando se realice una llamada a un método o a una función, no debe haber un espacio entre el nombre del método o la función y el paréntesis de apertura, no debe haber un espacio después del paréntesis de apertura, y no debe haber un espacio antes del paréntesis de cierre. En la lista de argumentos, no debe haber espacio antes de cada coma y debe haber un espacio después de cada coma.

**<?php**

bar();

$foo**->**bar($arg1);

Foo**::**bar($arg2, $arg3);

La lista de argumentos puede dividirse en múltiples líneas, donde cada una se indenta una vez. Cuando esto suceda, el primer argumento debe estar en la línea siguiente, y debe haber sólo un argumento por línea.

**<?php**

$foo**->**bar(

$argumentoLargo,

$argumentoMaslargo,

$argumentoTodaviaMasLargo

);

Estructuras de Control

Las reglas de estilo para las estructuras de control son las siguientes:

* Debe haber un espacio después de una palabra clave de estructura de control.
* No debe haber espacios después del paréntesis de apertura.
* No debe haber espacios antes del paréntesis de cierre.
* Debe haber un espacio entre paréntesis de cierre y la llave de apertura.
* El cuerpo de la estructura de control debe estar indentado una vez.
* La llave de cierre debe estar en la línea siguiente al final del cuerpo.

El cuerpo de cada estructura debe estar encerrado entre llaves. Esto estandariza el aspecto de las estructuras y reduce la probabilidad de añadir errores como nuevas líneas que se añaden al cuerpo de la estructura.

Sentencias If, elseif, else

Una estructura if tendrá el siguiente aspecto. Fíjese en el lugar de los paréntesis, los espacios y las llaves; y que else y elseif están en la misma línea que las llaves de cierre del cuerpo anterior.

**<?php**

**if** ($expr1) {

*// if cuerpo*

} **elseif** ($expr2) {

*// elseif cuerpo*

} **else** {

*// else cuerpo;*

}

La palabra clave elseif debería ser usada en lugar de else if de forma que todas las palabras clave de la estructura estén compuestas por palabras de un solo término.

Sentencias Switch, case

Una estructura switch tendrá el siguiente aspecto. Fíjese en el lugar donde están los paréntesis, los espacios y las llaves. La palabra clave case debe estar indentada una vez respecto al switch y la palabra clave break o cualquier otra palabra clave de finalización debe estar indentada al mismo nivel que el cuerpo del case. Debe haber un comentario como // no break cuando hay case en cascada no vacío.

**<?php**

**switch** ($expr) {

**case** 0**:**

**echo** 'Primer case con break';

**break**;

**case** 1**:**

**echo** 'Segundo case sin break en cascada';

*// no break*

**case** 2**:**

**case** 3**:**

**case** 4**:**

**echo** 'Tercer case; con return en vez de break';

**return**;

**default:**

**echo** 'Case por defecto';

**break**;

}

Sentencias While, do while

Una instrucción while tendrá el siguiente aspecto. Fíjese en el lugar donde están los paréntesis, los espacios y las llaves.

**<?php**

**while** ($expr) {

*// cuerpo de la estructura*

}

Igualmente, una sentencia do while tendrá el siguiente aspecto. Fíjese en el lugar donde están los paréntesis, los espacios y las llaves.

**<?php**

**do** {

*// cuerpo de la estructura;*

} **while** ($expr);

Sentencias For

Una sentencia for tendrá el siguiente aspecto. Fíjese en el lugar donde aparecen los paréntesis, los espacios y las llaves.

**<?php**

**for** ($i **=** 0; $i **<** 10; $i**++**) {

*// cuerpo del for*

}

Sentencias Foreach

Un sentencia foreach tendrá el siguiente aspecto. Fíjese en el lugar donde aparecen los paréntesis, los espacios y las llaves.

**<?php**

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

*// cuerpo foreach*

}

Sentencias Try, catch

Un bloque try catch tendrá el siguiente aspecto. Fíjese en el lugar donde aparecen los paréntesis, los espacios y los llaves.

**<?php**

**try** {

*// cuerpo del try*

} **catch** (PrimerTipoDeExcepcion $e) {

*// cuerpo catch*

} **catch** (OtroTipoDeExcepcion $e) {

*// cuerpo catch*

}

Closures

* Las closures deben declararse con un espacio después de la palabra clave function, y un espacio antes y después de la palabra clave use.
* La llave de apertura debe ir en la misma línea, y la llave de cierre debe ir en la línea siguiente al final del cuerpo.
* No debe haber un espacio después del paréntesis de apertura de la lista de argumentos o la lista de variables, y no debe haber un espacio antes del paréntesis de cierre de la lista de argumentos o la lista de variables.
* En la lista de argumentos y la lista variables, no debe haber un espacio antes de cada coma, y debe haber un espacio después de cada coma.
* Los argumentos de las closures con valores por defecto, deben ir al final de la lista de argumentos.
* Una declaración de una closure tendrá el siguiente aspecto. Fíjese en el lugar donde aparecen los paréntesis, las comas, los espacios y las llaves.

**<?php**

$closureConArgumentos **=** **function** ($arg1, $arg2) {

*// cuerpo*

};

$closureConArgumentosYVariables **=** **function** ($arg1, $arg2) **use** ($var1, $var2) {

*// cuerpo*

};

La lista de argumentos y la lista de variables pueden ser divididas en múltiples líneas, donde cada nueva línea se indentará una vez. Cuando esto suceda, el primer elemento de la lista debe ir en una nueva línea y debe haber sólo un argumento o variable por línea.

Cuando la lista de argumentos o variables se divide en varias líneas, el paréntesis de cierre y la llave de apertura deben estar juntos en su propia línea separados por un espacio.

A continuación se muestran ejemplos de closures con y sin lista de argumentos y variables, así como con listas de argumentos y variables en múltiples líneas.

**<?php**

$listaLargaDeArgumentos\_sinVariables **=** **function** (

$argumentoLargo,

$argumentoMasLargo,

$argumentoMuchoMasLargo

) {

*// cuerpo*

};

$sinArgumentos\_listaLargaDeVariables **=** **function** () **use** (

$variableLarga1,

$variableMasLarga2,

$variableMuchoMasLarga3

) {

*// cuerpo*

};

$listaLargaDeArgumentos\_listaLargaDeVariables **=** **function** (

$argumentoLargo,

$argumentoMasLargo,

$argumentoMuchoMasLargo

) **use** (

$variableLarga1,

$variableMasLarga2,

$variableMuchoMasLarga3

) {

*// cuerpo*

};

$listaLargaDeArgumentos\_listaDeVars **=** **function** (

$argumentoLargo,

$argumentoMasLargo,

$argumentoMuchoMasLargo

) **use** ($var1) {

*// cuerpo*

};

$listaDeArgumentos\_listaLargaDeVariables **=** **function** ($arg) **use** (

$variableLarga1,

$variableMasLarga2,

$variableMuchoMasLarga3

) {

*// cuerpo*

};

Fíjese que las reglas de formateo se aplican también cuando una closure se usa directamente en una función o llamada a método como argumento.

**<?php**

$foo**->**bar(

$arg1,

**function** ($arg2) **use** ($var1) {

*// cuerpo*

},

$arg3

);

Comentarios

Los programas PHP pueden tener dos tipos de comentarios: Comentarios de una sola línea (de dos formas) y comentarios multilineas. Los comentarios de una sola línea son aquellos que se encuentran delimitados por // y #. Los comentarios multilineas son aquellos que se encuentran delimitados por /\*...\*/.

Los comentarios son escrituras fueras de la programación, lo que significa que el interprete de PHP no ejecutara nada sobre esas líneas, las cuales tienen como única utilidad especificar al programador en turno ciertas instrucciones, anotaciones o notas mentales de quien hizo el código.

Formatos de los tipos de comentarios

Comentarios de una sola línea

Deben tener el siguiente formato:

*//Comentario de una linea, la siguiente linea es leida como codigo de programacion*

*#Comentario de una linea, tiene la misma utilidad que "//".*

Comentarios multilineas

Deben tener el siguiente formato:

*/\* Comentario multilinea*

*es capaz de abarcar muchas lineas,*

*posee una etiqueta de apertura y otra de cierre \*/*

Convenciones del código PHP

Referencias:

<http://coppeldev.github.io/php/standards/coppel.html>

<https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_PHP#PHP_b%C3%A1sico>