***Guía de estilo de codificación***

Esta guía amplía y extiende el estándar de codificación básica PSR-1.

La objetivo de esta guía es la de reducir la dificultad cuando se lee código de diferentes autores. Lo realiza mediante la enumeración de una serie de reglas común y expresiones sobre cómo dar formato al código PHP.

En el documento original se usa el RFC 2119 para el uso de las palabras MUST, MUST NOT, SHOULD, SOULD NOT y MAY. Para que la traducción sea lo más fiel posible, se traducira siempre MUST como el verbo deber en presente (DEBE, DEBEN), SHOULD como el verbo deber en condicional (DEBERÍA, DEBERÍAN) y el verbo MAY como el verbo PODER.

**1. Visión general**

El código DEBE seguir el estándar PSR-1.

El código DEBE usar 4 espacios como indentación, no tabuladores.

NO DEBE haber un límite estricto en la longitud de la línea; el límite DEBE estar en 120 caracteres; las líneas DEBERÍAN tener 80 caracteres o menos.

DEBE haber una línea en blanco después de la declaración del namespace, y DEBE haber una línea en blanco después del bloque de declaraciones use.

Las llaves de apertura de las clases DEBEN ir en la línea siguiente, y las llaves de cierre DEBEN ir en la línea siguiente al cuerpo de la clase.

Las llaves de apertura de los métodos DEBEN ir en la línea siguiente, y las llaves de cierre DEBEN ir en la línea siguiente al cuerpo del método.

La visibilidad DEBE estar declarada en todas las propiedades y métodos; abstract y final DEBEN estar declaradas antes de la visibilidad; static DEBE estar declarada después de la visibilidad.

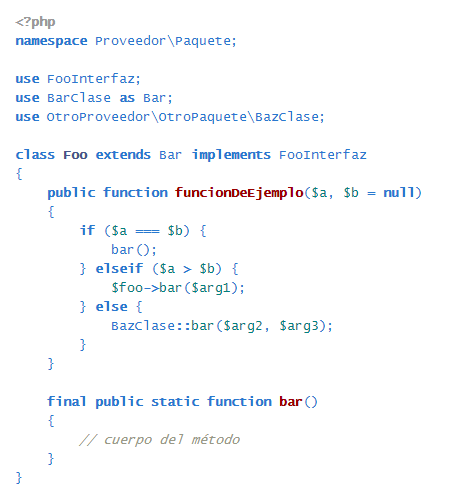
Las palabras clave de las estructuras de control DEBEN tener un espacio después de ellas, las llamadas a los métodos y las funciones NO DEBEN tenerlo.

Las llaves de apertura de las estructuras de control DEBEN estar en la misma línea, y las de cierre DEBEN ir en la línea siguiente al cuerpo.

Los paréntesis de apertura en las estructuras de control NO DEBEN tener un espacio después de ellos, y los paréntesis de cierre NO DEBEN tener un espacio antes de ellos.

**1.1. Ejemplo**

Este ejemplo incluye algunas de las siguientes reglas a modo de visión general rápida:



**2. General**

**2.1 Codificación estándar básica**

El código DEBE seguir las normas expuestas en el estándar PSR-1.

**2.2 Ficheros**

Todos los ficheros PHP DEBEN usar el final de línea Unix LF.

Todos los ficheros PHP DEBEN terminar con una línea en blanco.

La etiqueta de cierre ?> DEBE ser omitida en los ficheros que sólo contengan código PHP.

**2.3. Líneas**

NO DEBE haber un límite estricto en la longitud de la línea.

El límite flexible de la línea DEBE estar en 120 caracteres; los correctores de estilo automáticos DEBEN advertir de ésto, pero NO DEBEN producir errores.

Las líneas NO DEBERÍAN ser más largas de 80 caracteres; las líneas más largas de estos 80 caracteres DEBERÍAN dividirse en múltiples líneas de no más de 80 caracteres cada una.

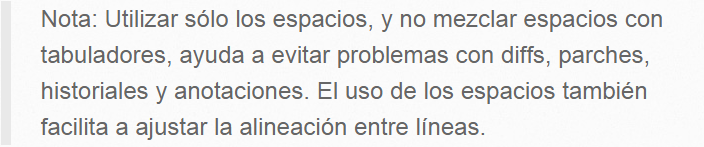
NO DEBE haber espacios en blanco al final de las líneas que no estén vacías.

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.

**2.4. Indentación**

El código DEBE usar una indentación de 4 espacios, y NO DEBE usar tabuladores para la indentación.



**2.5. Palabras clave y true/false/null.**

Las Palabras clave de PHP DEBEN estar en minúsculas.

Las constantes de PHP true, false y null DEBEN estar en minúsculas.

**3. Espacio de nombre y declaraciones use**

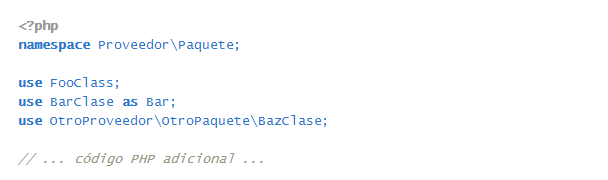
Cuando esté presente, DEBE haber una línea en blanco después de la declación del namespace.

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:



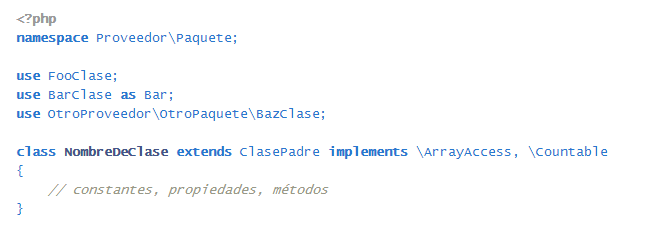
**4. Clases, propiedades y métodos**

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

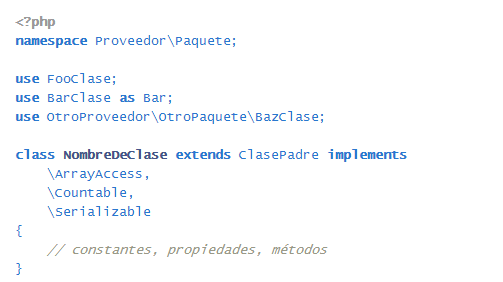
**4.1. Extensiones e implementaciones**

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.



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.



**4.2. Propiedades**

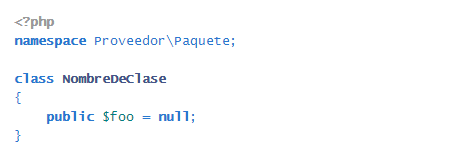
La visibilidad DEBE ser declarada en todas las propiedades.

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 propiedas tendrá el siguiente aspecto.



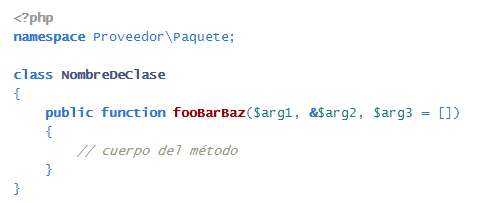
**4.3. Métodos**

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.

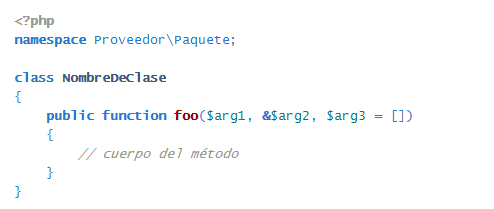
La declaración de un método tendrá el siguiente aspecto. Fíjese en la situación de los paréntesis, las comas, los espacios y las llaves:



**4.4. Argumentos de los métodos**

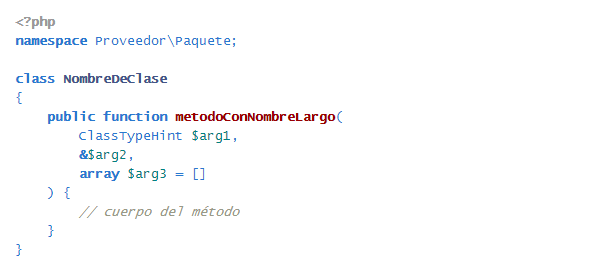
En la lista de argumentos NO DEBE haber un espacio antes de cada coma y DEBE haber un espacio después de cada coma.

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



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.

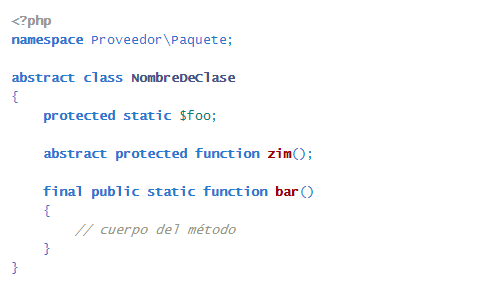
Cuando la lista de argumentos se divide en varias líneas, el paréntesis de cierre y la llave de apertura DEBEN estar juntos en su propia línea separados por un espacio.



**4.5. abstract, final, y static**

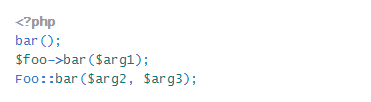
Cuando estén presentes las declaraciones abstract y final, DEBEN preceder a la declaración de visibilidad.

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

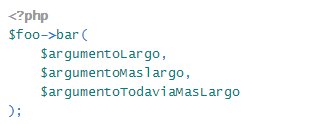


**4.6. Llamadas a métodos y funciones**

Cuando se realize 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.



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.



**5. Estructuras de control**

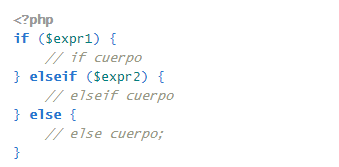
Las reglas de estilo para las estructuras de control son las siguientes:

* 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.

**5.1. if, elseif, else**

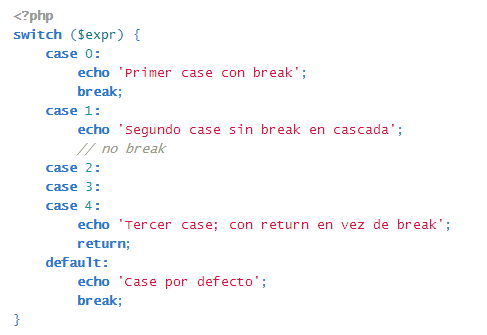
Una estructura if tendrá el siguiente aspecto. Fíjese en el lugar de los paréntesis, los espacios y las llaves; y que else y elseif están en la misma línea que las llaves de cierre del cuerpo anterior.



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.

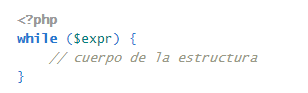
**5.2. switch, case**

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 indentadas al mismo nivel que el cuerpo del case. DEBE haber un comentario como // no break cuando hay case en cascada no vacío.

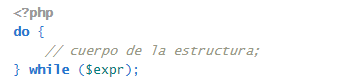


**5.3. while, do while**

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

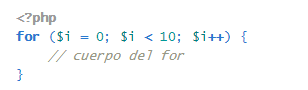


Igualmente, una sentencia do while tendrá el siguiente aspecto. Fíjese en el lugar donde están los paréntesis, los espacios y las llaves.



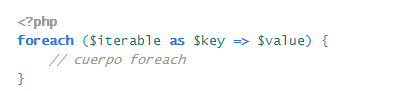
**5.4. for**

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



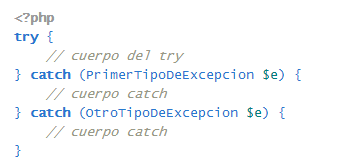
**5.5. foreach**

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



**5.6. try, catch**

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



**6. Closures**

Las closures DEBEN declararse con un espacio después de la palabra clave function, y un espacio antes y después de la parabra clave use.

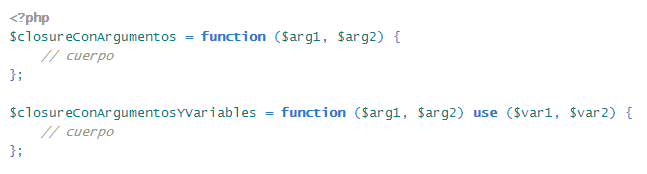
La llave de apertura DEBE ir en la misma línea, y la llave de cierre DEBE ir en la línea siguiente al final del cuerpo.

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 QUE 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.

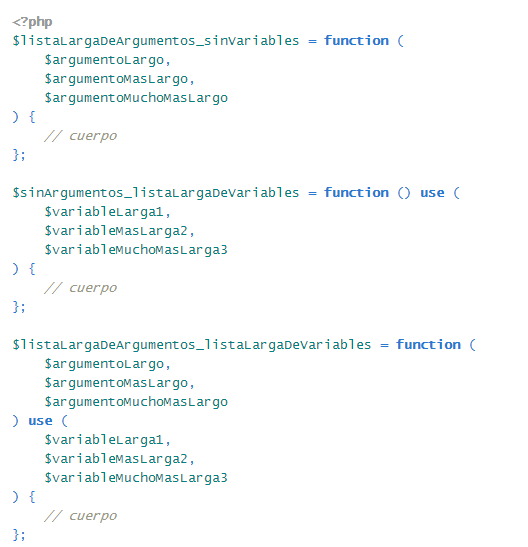
Una declaración de una closure tendrá el siguiente aspecto. Fíjese en el lugar donde aparecen los paréntesis, las comas, los espacios y las llaves.

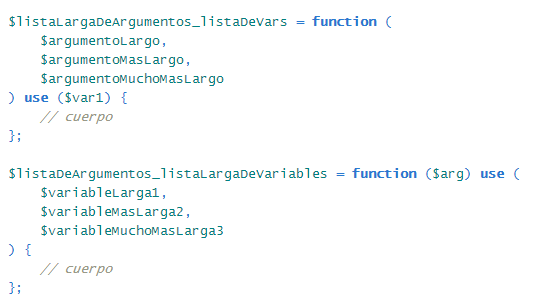


La lista de argumetos y la lista de variables PUEDEN ser divididas en múltiples líneas, donde cada nueva línea se indentará una vez. Cuando esto suceda, el primer elemento de la lista DEBE ir en una nueva línea y DEBE haber sólo un argumento o variable por línea.

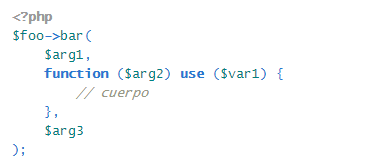
Cuando la lista de argumentos o variables se divide en varias líneas, el paréntesis de cierre y la llave de apertura DEBEN estar juntos en su propia línea separados por un espacio.

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.





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



**Notas:**

**Indentado:** Unas líneas que empiezan con uno o más espacios a principio de línea, indican un texto indentado (<ul>). Cada dos espacios se incrementa de un nivel la indentación.