

1ª versión: 11/08/2009

Última revisión: 11/08/2009

Versiones utilizadas:

SO: UBUNTU 9.04

**Resumen:**

Este documento es un resumen del PHP5 orientado a programadores que ya conozcan otros lenguajes y los sistemas Unix-Linux.

**Reglas generales de codificación:**

- Marcadores de inicio y final de scrip: "`<?php`" . . . "`?>`"
- Utilizar "TABS" no espacios en blanco.
- Incluir cabecera con información del fichero, versión, autor, explicaciones, etc.
- Comentar las funciones ANTES de su declaración. El comentario ha de ser claro y conciso y suficiente para entender que es lo que hace la función.
- Nombres de los elementos: concisos y descriptivos. Preferible una sola palabra. Utilizar el subrayado "\_" como separador. Máximo 4 palabras en un nombre.
- Variables, funciones y argumentos: Nombre en minúsculas.
- Nombre de las Funciones: verbos en infinitivo.
- Clases: incluir el código de cada clase en un fichero ".php" independiente y cuyo nombre sea = al de la clase. El nombre de una clase tendrá el primer carácter en mayúscula.
- Poner los corchetes en línea independiente.
- Poner un espacio a cada lado de los símbolos binarios y a un lado en los símbolos unarios:  
Ejemplo: `for ($i = 5; $i <= $j; $i++ )`
- Utilizar los paréntesis para asegurar la precedencia de los operadores (y facilitar la lectura).

**Comentarios:**

- `/*` Estilo C - Multi-línea hasta el conjunto de fin `*/`
- `//` Estilo C++ - Solo hasta fin de línea...
- `#` Estilo shell – Solo hasta fin de línea...
- Cuidado, los comentarios PHP han de estar siempre dentro del *script* "`<?php ... ?>`"
- Recordatorio: Comentarios en HTML: `<!-- ... -->`

**Constantes:**

- Se recomienda nombre de las constantes en mayúsculas.
- Declaradas mediante sentencia "define". Ejemplos:

```
define ("PI", 3.141592);
```

```
define ("EMPRESA", "ACME, S.A.");
```

Función "`constant()`" retorna el valor de una constante. También funciona la sentencia "`echo`".

Ejemplos:

```
echo EMPRESA; Resultado "ACME, S.A."
```

```
echo constant ("EMPRESA") ; Resultado idéntico.
```

Función "`defined()`" retorna cierto si la constante ya ha sido definida.

Algunas constantes predefinidas:

- PHP\_VERSION
- PHP\_OS

- PHP\_LIBDIR
- ...

Constantes “mágicas”:

- `__LINE__` Número de la línea en que se encuentra la constante `__LINE__`
- `__FILE__` Ruta completa del fichero PHP que se ejecuta.
- `__FUNCTION__` Nombre de la función donde se encuentra la constante...
- `__CLASS__` Nombre de la clase.
- `__METHOD__` Nombre del método.

### Variables:

Asignación dinámica de tipo sin declaración previa. Es muy aconsejable inicializar cada variable antes de su utilización. Reasignan dinámica (peligro !!!)

Tipos:

- Entero (integer); Admite formato octal ( `$var = 042;` ) y hexa ( `$var = 0x42` ). La sentencia “echo” mostrará el valor decimal de la variable. Para imprimir formatos octal y hexa se utiliza la función “printf”.
- Coma flotante (double);
- Cadena (string) → “...”; '...';
- Booleano (True # 0 ; False = 0);
- Null; NULL; null ;
- Vector (array)
- Objeto (object);

Nombre de las variables:

- Se inicia siempre su nombre con un signo dólar “\$”.

- No puede empezar por una cifra
- Puede empezar por un subrayado o un carácter alfabético.
- Diferencia entre mayúsculas y minúsculas.

**Variables de variables:**

`$A = "Hola"` ; Asignación al espacio de memoria denominado "A".

`$$A = Mundo` ; Asignación al espacio de memoria denominado "Hola".

`echo ("$A $Hola")` ; Resultado: "Hola Mundo".

`Echo ("$A $$A")` ; Resultado: "Hola Mundo". (mismo resultado).

**Variables tipo vector (array):**

Admiten dos tipos de índices:

- índice numérico (tradicional). 1a. Clave = 0.
- índice simbólico: una cadena de identificación del elemento.
- Permite la indexación automática. Ejemplo: `$dia= array( "Lunes", "Martes", "Miércoles" ...)`;  
- siendo `$dia[1]` igual a "Martes".
- Índice asociativo. Ejemplo: `$capital = array( "ES" => "Madrid", "FR" => "París", "GB" => "Londres", ...)`; - siendo `$capital["FR"]` igual a "París". Idem `$capital[1]`.

Funciones para recorrido y posicionamiento dentro de un vector (array):

- `count( vector )` ; - Retorna el número de elementos del vector.
- `reset( vector )` ; - Saltar al primer elemento.

- `end( vector )` ; - Saltar al último elemento.
- `next( vector )` ; - Avanzar un elemento dentro del vector.
- `prev( vector )` ; - retroceder un elemento dentro del vector.
- `current( vector )` ; - valor (contenido) del elemnto actual.
- ...

**Cadenas de caracteres:****1.- Comillas simple y doble:**

Si `$valor = 123`;

- `echo ('Valor = $valor \t Hola');` tiene como resultado “Valor = \$valor \t Hola”.
- `echo (“Valor = $valor \t Hola”);` tiene como resultado “Valor = 123        Hola”.
- Comilla doble: contenido interpretado por el compilador.
- Comillas simples: contenido NO interpretado por el compilador.
- Se recomienda utilizar siempre comillas simples: Ejemplo: `echo 'Valor = ' . $valor . ' Hola';`
- ...

**2.- Secuencias de escape:**

- `\$ ; \' ; \' ; \\ ; \n ; \r ; \t ;`

**Funciones para el manejo de las variables:**

- `isset($var)` ; Retorna cierto si la variable existe;
- `unset($var)` ; Libera la memoria y destruye la variable.
- `gettype($var)` ; retorna el tipo de la variable (integer; double; string; array; object; class, o unknow type).
- `settype($var, “tipo”)`; retorna cierto si ha podido convertir la variable al tipo especificado.
- `empty($var)`; retorna cierto si la variable está vacía, no existe o su valor es “0”.
- `is_integer($var)`; `is_double($var)`; `is_string($var)`; retornan cierto si el tipo es correcto
- `intval($val)`; `doubleval($var)`; `strval($var)`; retornan el valor convertido de tipo (¿o NULL si no pueden convertirlo?).

**Funciones:**

- Declaración mediante sentencia “**function**”. Ejemplo:

```
function nombre_funcion( $param1, $param2, ... ) {
```

```
    cuerpo de la función;
```

## NT-03

## Resumen PHP5 para programadores

```

return( valor de retorno);

}

```

- Paso de parámetros por valor (\$param)
- Paso de parámetro por referencia (añadir carácter "&" delante del "\$" del nombre del parámetro: &\$param).
- 

**Operadores:**

|       |                           |    |                       |    |                       |
|-------|---------------------------|----|-----------------------|----|-----------------------|
| '='   | Asignación                | && | And                   | +  | Suma                  |
| '=='  | Comparación               |    | Or                    | -  | Resta                 |
| '===' | Identidad de valor y tipo | ^  | Xor                   | *  | Producto              |
| '!='  | Distinto                  | !  | Not                   | /  | División              |
| <>    | Distinto                  |    |                       | %  | Módulo                |
| >     | Mayor                     | >= | Mayor o igual         | ++ | Suma unária           |
| <     | Menor                     | <= | Menor o igual         | -- | Resta unária          |
| .     | Concatenación cadenas     | >> | Desplazamiento de bit | << | Desplazamiento de bit |
| `...` | Ejecución orden           |    |                       |    |                       |

Notas sobre la suma y resta unarias:

|  |  |
|--|--|
| \$a = 5;                                 | \$a = 5;                                 |
| \$b = \$a++; Resultado: \$a = 6; \$b = 5 | \$b = ++\$a; Resultado: \$a = 6; \$b = 6 |

## NT-03

## Resumen PHP5 para programadores

Expresiones válidas:

|                          |                               |                                 |                                      |
|--------------------------|-------------------------------|---------------------------------|--------------------------------------|
| <code>\$var++</code>     | Tras la asignación            | <code>\$a &amp;= \$b;</code>    | <code>\$a = \$a &amp; \$b;</code>    |
| <code>\$var--</code>     |                               | <code>\$a  = \$b;</code>        | <code>\$a = \$a   \$b;</code>        |
| <code>++\$var</code>     | Antes de la asignación        | <code>\$a ^= \$b;</code>        | <code>\$a = \$a &amp; \$b;</code>    |
| <code>--\$var</code>     |                               | <code>\$a &gt;&gt;= \$b;</code> | <code>\$a = \$a &gt;&gt; \$b;</code> |
| <code>\$a += \$b;</code> | <code>\$a = \$a + \$b;</code> | <code>\$a &lt;&lt;= \$b;</code> | <code>\$a = \$a &lt;&lt; \$b;</code> |
| <code>\$a -= \$b;</code> | <code>\$a = \$a - \$b;</code> |                                 |                                      |
| <code>\$a *= \$b;</code> | <code>\$a = \$a * \$b;</code> |                                 |                                      |
| <code>\$a /= \$b;</code> | <code>\$a = \$a / \$b;</code> |                                 |                                      |
| <code>\$a %= \$b;</code> | <code>\$a = \$a % \$b;</code> |                                 |                                      |
| <code>\$a .= \$b;</code> | <code>\$a = \$a . \$b;</code> |                                 |                                      |

**Recomendaciones:**

- Operadores unarios en una línea independiente (siempre que sea posible).

**Operador de Ejecución de orden:**Ejemplo: `$listado = `ls -l` ;`**Operador Ternario (sentencia condicional):**

- Para utilizar solo en casos simples y en una sola línea.
- Formato:

*Expresión1* Evaluación Lógica *Expresión2* ? *Sentencia1* : *Sentencia2* ;

- Ejemplo:

$$\$min = (\$A < \$B ? \$A : \$B) ;$$

**Operador de supresión de errores en una función:**

Se añade una arroba al nombre de la función cuando se la llama. Ejemplo:

```
@imprimir(arg_1, arg_2);
```

Si al ejecutarse la función se produce algún error, no se mostrará y la ejecución prosigue, excepto si se trata de un error muy grave que provoque el aborto de la ejecución.

**Estructuras de programación:****Codicional IF:**

```
if ( condición_lógica) {  
    sentencias ;  
}  
else {  
    sentencias ;  
}
```

Nota: La segunda parte (else { ... } ) es opcional.

**Multiselección SWITCH:**

```
Switch (expresion) {  
    case valor1 :  
        sentencias ;  
        break ;  
    case valor2 :  
        sentencias ;
```

```
        break ;  
  
...  
  
    default :  
  
        sentencias ;  
  
}
```

Nota: si no se incluye la sentencia “break” se continua evaluando los valores que cumplan con la condición...

**Bucle WHILE:**

```
while (condición) {  
    sentencias;  
}
```

**Bucle DO WHILE:**

```
do {  
    sentencias;  
} while (condición) ;
```

**Bucle FOR:**

```
for (expresión inicial; condición de fin; expresión de bucle ) {
```

```
sentencias; }
```

Regla recomendada para los corchetes: *1 corchete en 1 línea y 1 línea para cada corchete!*

**Bucle FOREACH:**

```
foreach( $vector as $valor ) { // Para cada uno de los valores del vector...
```

```
    sentencias con referencia a $valore;
```

```
}
```

Alternativa:

```
foreach( $vector as $clave => $valor) { // Para cada uno de los valores del vector...
```

```
    sentencias con referencias $clave y $valor; // Muy util en el tratamiento de formularios...
```

```
}
```

**Bucle WHILE – EACH:**

```
while ($variable = each ( $vector)) {
```

```
    sentencias ;
```

```
}
```

Notas: El elemento de índice 0 ( $\$variable(0)$ ) corresponde al índice simbólico (cadena) del elemento del vector y el índice 1 al valor del elemento del vector.

Ejemplo:

## NT-03

## Resumen PHP5 para programadores

```

<?php
    $alumno["Nombre"] = "Pedro" ;
    $alumno["Apellidos"] = "Sanchez Rodriguez" ;
    $alumno["edad"] = 14 ;

    while ( $un_alumno = each( $alumno ) {
        echo ("$un_alumno[ 0 ] = $un_alumno[ 1 ]" . "<br />" ) ;
    }
?>

```

Resultado:

Nombre = Pedro

Apellidos = Sanchez Rodriguez

Edad = 14

### Funciones de tratamiento de cadenas (strings):

|                                  |  |            |                                      |
|----------------------------------|--|------------|--------------------------------------|
| echo(...)                        |  | chr(valor) | Carácter ascii del valor             |
| print(...)                       |  | ord(c)     | Valor decimal del carácter ascii "c" |
| printf(...)                      |  | trim(...)  | Elimina espacios en blanco           |
| print_r(...)                     |  | ltrim(...) | Elimina espacios a la izquierda      |
| strlen(c)                        | Longitud cadena c  | rtrim()    | Elimina espacios a la derecha.       |
| str_replace(s1, s2, cadena)      | Substituye s2 por s1 en cadena.  |            |                                      |
| strtolower(cadena)               | Cadena en minúsculas   |            |                                      |
| strtoupper(cadena)               | Cadena en mayúsculas   |            |                                      |
| substr(cadena, inicio [, lmax] ) | Fragmento de la cadena a partir de la posición "inicio". Opcional: "lmax" = longitud máxima. |            |                                      |
| strchr(cadena, c)                | Subcadena de cadena a partir del carácter "c"  |            |                                      |
| strpos(s1, cadena [, pos])       | Posición de la cadena s1 en cadena. Opcional: pos = posición inicio de la búsqueda.          |            |                                      |

## NT-03

## Resumen PHP5 para programadores

|                      |   |
|----------------------|---|
| strcmp(cad_1, cad_2) | Comparación entre cad_1 y cad_2.                          |
| explode(c, cadena)   | Descompone cadena en subcadenas, siendo "c" el separador. |

## Otras funciones:

|  |   |
|--|---|
| round(valor, decimales)                          | Redondeo                                      |
| sprintf(cadena_formato, \$variable/s)            | Retorna una cadena con el formato deseado.    |
| mail(e-amil destino, asunto, mensaje, remitente) | Retorna Cierto o Falso.                       |
| Getdate()  | Retorna la fecha del sistema en un vector     |
| Time()   | Retorna los segundos desde 1/1/1970           |
| date(c_formato, valor_time)                      | Cadena de la fecha y/o hora (según c_formato) |

```
// Ejemplo de función sprintf()
<?php
    $precio_bruto = 1234 ;
    $iva = 0.16 ;
    $iva = $iva * $precio_bruto ;
    $total = $precio_bruto + $iva ; // $total es ahora de tipo "double"
    $total = sprintf("%01.2f", $total) ; // Cambio de tipo a cadena
    echo "$precio_bruto euros mas $iva euros de IVA serán $total euros."
?>
```

**Ejemplo de manejador de errores:**

```
// Función de visualización de los errores (error_reporting).
<?php
    function trata_errores($error, $mensaje, $fichero, $linea)
    {
        echo "<b>::ERROR::</b> <br />";
        echo "Se ha producido un error de tipo: $error - $mensaje en la línea
$linea del fichero $fichero" ;
    }
?>
// Declaración del gestor de errores a utilizar...
<?php
    set_error_handler("trata_errores");
    // resto de código PHP...
?>
```

```
// Ejemplo: dibujar una tabla HTML
<?php
function dibuja_tabla( $n_filas, $n_columnas) {
    echo ('<div class="tabla">');
    echo ('<table>');
        for ($fila = 0; $fila < $n_filas; $fila++) {
            echo ('<tr>');
            for ($col = 0; $col < $n_columnas; $col++) {
                echo ('<td>' . $fila . ', ' . $col . '</td>');
            }
            echo ('</tr>');
        }
        echo ('</table>');
    echo ('</table>');
}
?>

// Llamada ala función...
<?php
    dibujar_tabla (5, 3);
?>
```

Configuración – Fichero [php.ini]:

Pueden existir diversos ficheros de configuración con el nombre [php.ini]. Los + significativos son:

1. **/etc/php5/apache2/php.ini** – Configuración para la ejecución del código PHP embebido en HTML e interpretado por el servidor web Apache 2 como módulo.
2. **/etc/php5/cgi/php.ini** – Configuración para la ejecución del código PHP embebido en HTML e interpretado como CGI por el servidor web Apache 2.
3. **/etc/php5/cli/php.ini** – Configuración de para la ejecución CLI (*Comand Line Interface* = Línea

de ordenes).

4. Otros ficheros [php.ini]: El interprete PHP lee, si existen, los ficheros de configuración [php.ini] en el siguiente orden:
  1. Directorio actual de trabajo.
  2. Directorios especificados en la variable de entorno PHPRC.
  3. Directorio especificado en la compilación del interprete (en principio uno de los 3 anteriores, en función del interprete utilizado).

#### Sintaxis del fichero [php.ini]

- Comentarios:
  - líneas que se inician en punto y coma “;”. De hecho, en cualquier línea, a partir de un punto y coma el resto de la línea se considera comentario.
  - Etiquetas de sección [Ejemplo] (se ignoran).
- Directivas:
  - Se diferencian mayusculas / minusculas
  - Asignación de valor mediante signo “=”. Ejemplo: **memory\_limit = 128M**
  - Valores posibles: una cadena; un número; una constante PHP (Ejemplo: E\_ALL); una de las constantes básicas (On, Off, True, False, Yes, No, None) o una expresión lógica (or = |; and = &; Not = !; Not = ~; Ejemplo: E\_ALL & ~E\_NOTICE);
  - Valores booleanos: (True; 1; Yes; On) ; (False; 0; No; Off)
  - Una cadena vacia se puede expresar Valor = ; Valor = “”; o bien como Valor = none

Algunas directivas en [php.ini]

- **short\_open\_tag** = [On/Off] ; Permite el formato corto (desaconsejado) de inicio de script PHP (“<?” en vez de “<?php”. Recomendación: Poner en “Off”).
- **disable\_funtions** = [lista] ; Permite desactivar determinadas funciones (red, correo, etc.) potencialmente peligrosas.
- **max\_execution\_time** = 30 ; Tiempo máximo, en segundos, de ejecución de un script PHP en el servidor.
- **memory\_limit** = 64M ; Tamaño máximo de ocupación de memoria por los scrips PHP. Poner al máximo posible...(128M no está mal para empezar ;-)
- **error\_log** = [On / Off] ; Crear o no reporte de los errores.
- **error\_reporting** = E\_ALL & ~E\_NOTICE ; Nivel de errores que son reportados. Ver lista de opciones (definidas mediante constantes) en el propio fichero [php.ini].
- **register\_globals** = [On / Off] ; Declara todas las variables como globales. Recomendación: Poner en “Off”.
- **magic\_quotes\_runtime** = [On / Off] ; Añadir el símbolo de ESC cuando se pasan parametros entre comillas. (?)
- **file\_uploads** = [On / Off] ; Permite (o no) subir ficheros (¿mediante formularios?).
- **upload\_max\_filesize** = 2M ; Tamaño máximo para los ficheros subidos por scripts de PHP.
- **include\_path** = "./usr/share/php" ; Añade rutas de bibliotecas de otros scripts.
- **auto\_append\_file** “xxx” ; Código PHP que se ejecutará tras cada ejecución (por ejemplo para un pie de página).
- **auto\_prepend\_file** “xxx” ; Código PHP que se ejecutará antes de toda ejecución (para incluir

funciones o declaraciones de constantes, por ejemplo).

Notas:

- En [/usr/share/doc/php5-common/examples] hay ejemplos de ficheros [php.ini].
- Para que APACHE se perciba de las modificaciones en el fichero [php.ini] es necesario reinicializarlo.
- Algunas directivas pueden ser modificadas desde dentro del propio script mediante la función **ini\_set(...)**.
- La función **ini\_get(...)** permite capturar el valor de una directiva desde dentro del script.

## Fuentes de información:

- Código libre (Inglés): <http://www.codewalkers.com>
- Revista PHPsolutions (Inglés): <http://www.phpsolmag.org>
- Comunidad PHP de Extremadura (Castellano): <http://www.sinuh.org>
- Comunidad de programadores en PHP: <http://www.phpbuilder.com>
- ...