# PHP 7: El cambio en PHP

Eduardo José Ávila Portillo*

**Resumen**:
En el presente artículo se analiza las características de la última versión de php, sus cambios en relación a versiones anteriores, el impacto que representa sobre plataformas importantes basadas en PHP y la programación web.

**Abstract** :
In this article we analyze the features of PHP version 7.0, the changes in relation to past versions, the impact on PHP based platforms and the web programming.

## 1. Introducción

Según php.net[1], PHP es un lenguaje libre para programación web, popularizado en masa desde el 2000, en su versión 4.0, pero creada por Rasmus Lerdorf en el año de 1994, con el objetivo de ejecutar tareas pequeñas tal como lo prueba su nombre PHP: "Personal Home Pages". Siglas que tras la constante mejora de su kernel y aumento de potencia terminan desfasadas por el acrónimo recursivo PHP:"PHP Hypertext Preprocessor" en su versión 3.0; desde entonces suma seguidores y críticas que lo obligan a mejorar. Como resultado tenemos la versión 7 de PHP, ofreciéndonos nuevas características algunas de ellas innovadoras en el campo de la programación web, como nos lo comenta Leroy Ley en su artículo[2] "PHP 7: Lo nuevo", publicado en mycyberacademy.com, una revista digital especalizada en desarrollo y tecnologia.

## 2. Comparación entre PHP 7 y versiones anteriores

### a. Renovación en el API

Rasmus Lerdorf, durante la ponencia "speeding up the web with PHP 7"[3] en San Francisco, U.S.A. nos comenta que PHP 7 cuenta con una API de ejecutables hechos en C; aprovechando así la velocidad de estos ensamblados, sin necesidad de hacer el embebido de ejecutables C/C++ mientras programamos, como antes estábamos acostumbrados a realizar.

Esto cierra un poco la brecha entre los desarrolladores tradicionales que usan código 100% PHP en sus páginas y los avanzados que incrustan ejecutables hechos en C/C++ para aumentar la velocidad de los procesos. Como conclusión ya no es necesario embeber ejecutables C/C++, php 7 ***duplica la velocidad*** de PHP5.6 con su API hecha en C, según Rasmus nos comenta[5].

### b. Retorno con tipos de datos

Las funciones en PHP ya no retornan un valor genérico, sino que se deberá asignar el tipo del valor que se devuelve, sin importar que sean valores primitivos u objetos de clases creadas por el usuario.

```
class C {}

function getC() : C
{
        return new C;
}

var_dump(getC());
```

* *Estudiante de Ingeniería en ciencias de la computación de la Universidad Don Bosco. Desarrollador web PHP en la Universidad Don Bosco (eduardo.avila@udb.edu.sv).*

El tipo de valor se coloca después del paréntesis de cierre precedido del carácter dos puntos (:).

### c. Operador de fusión de NULL

Según nos lo muestra Leroy en su artículo, el operador de fusión está compuesto por dos signos de interrogación y nos da más orden en el código php eliminando la necesidad de usar operadores ternarios.

> "Bueno el operador de fusión ofrece una alternativa, mismo objetivo, mismo resultado… más elegancia"

Se ocupa para saber si existe o no una variable, sin usar un if, sin usar el isset y a veces el operador de negación y sin usar el operador [condición]?[true] :[false].

```
if(!isset($‘var)){  die(‘Var no existe’);  }

//o sino, con operador ternario
(!isset($‘var))?die(‘Var no existe’):’’
```

**Código desfasado**

En PHP 7 la sintaxis será:

```
$var ?? die(‘Var no existe’);
```

**Código PHP 7**

Según Leroy, este operador es versátil e inclusive se puede concatenar. Como ejemplo digamos que queremos asignar valor a una variable desde otra recogida por GET, por POST o por otra variable y si no existe asignar cero; php7 lo resuelve así:

```
$a=$_POST[‘v’] ??$_GET[‘V’] ??$v ?? ‘0’
```

**Código PHP 7**

Problema resuelto en ¡¡Una sola línea de código!!

Lo que Leroy Ley no menciona, es la potencial segunda utilidad de este operador, y me refiero a la eliminación del uso desordenado del operador ‘or’. Normalmente un programador PHP utiliza el operador "or" para mandar un mensaje en caso una función retorne ‘false’. por ejemplo:

```
function retorna0(){return false;}

retorna0() or die("Función retorno false");

/* Imprime la frase ‘función retorno false’ */
```

**Código desfasado**

Pero con PHP 7 el código se hace más ordenado:

```
function retorna0(){return false;}

retorna0() ?? die("Función retorno false");

/* Imprime la frase ‘función retorno false’ */
```

**Código php 7**

Notamos más orden y de elegancia pues dejamos de abusar de las características básicas del operador lógico "or".

### d. Operador Spaceship

Podemos leer con amplio detalle una entrada hecha por php.net donde nos comenta los fines de este operador

> "This RFC adds a new operator for combined comparison. Similar to strcmp()"

como un signo de ‘menor que’, un signo de ‘igual’ y otro de ‘mayor que’ (<=>)El operador define ¿cuál valor es mayor, cual es menor o si son iguales?

```
$v1 = 100;
$v2 = 1;
$v  =  ($v1<$v2)?-1:(($v1>$v2)?1:0);
// 1

$v1 = 22;
$v2 = 9000;
$v  =  ($v1<$v2)?-1:(($v1>$v2)?1:0);
// -1

$v1 = 5;
$v2 = 5;
$v  =  ($v1<$v2)?-1:(($v1>$v2)?1:0);
// 0
```

**Código desfasado**

este código que implica 2 procesos con operadores ternarios, se ha visto desfasado por un solo operador como se muestra en el siguiente ejemplo:

```
$v1 = 100;
$v2 = 1;
$v = $v1<=>$v2; // devuelve 1
```

**Código PHP 7**

```
$v1 = 22;
$v2 = 9000;
$v = $v1<=>$v2; // devuelve -1

$v1 = 5;
$v2 = 5;
$v = $v1<=>$v2; // devuelve 0
```

**Código PHP 7**

### e. Todavía hay mas

Por su puesto hay muchos más cambios, pero son estos los que se consideraron más significativos al momento de trabajar con PHP. Arrays constantes, división entera, entre otras, son características que Leroy nos explica; pero considero que con las mencionadas hasta aquí se demuestra que PHP ha terminado una etapa y comienza otra.

## 3. PHP, factor de discriminación

Muchos programadores web, bajo la idea que PHP es un lenguaje desordenado y poco restrictivo, no lo prefieren; pero PHP 7 es una clara mejora en esa debilidad y una clara mejora en la eficiencia. Es mas las graficas lo demuestran, las aplicaciones hechas con PHP 7 mantienen más request por segundo que las hechas con las otras versiones. La próxima vez que elija una version de una plataforma basada en PHP para uso institucional, revisar con detalle la versión del PHP utilizado, la aplicación basada en PHP 7 es más eficiente para manejar la memoria RAM, Según afirma Rasmus Lerdorf, las mismas tareas son realizadas una cantidad menor de procesos; dado como resultado web application más ágiles que las desarrolladas con php 5.6. Como Lerdorf lo demuestra en las figuras 1, 2, 3 y 4

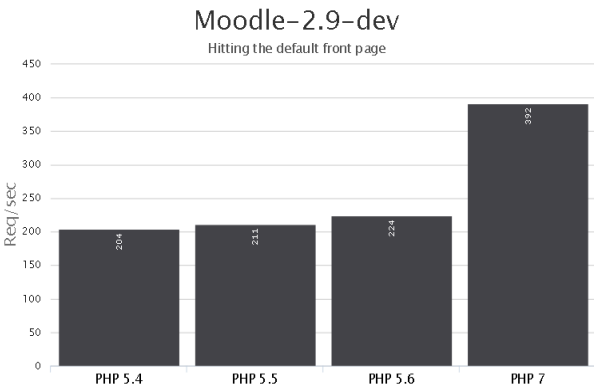


*Figura 1 – request por segundo que realiza Moodle2.9 dependiendo de la versión de PHP*

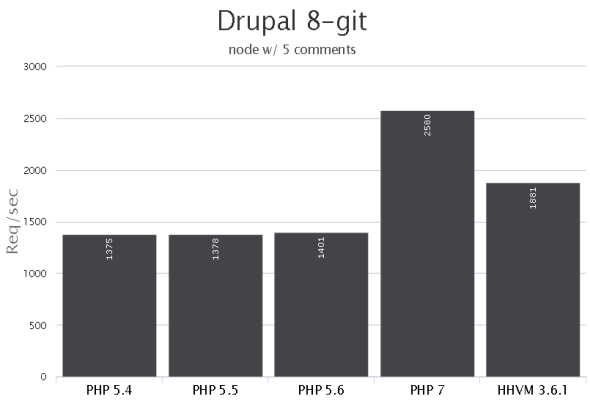


*Figura 2 – Request/segundo que realiza el CMS Drupal dependiendo de la versión de PHP*

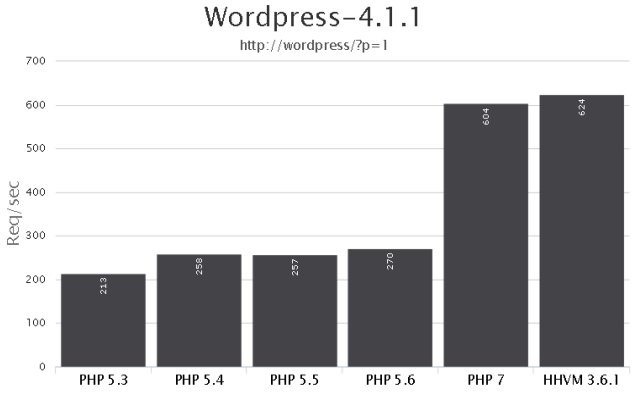


*Figura 3 – Request/segundo que realiza el CMS Wordpress dependiendo de la versión de PHP*

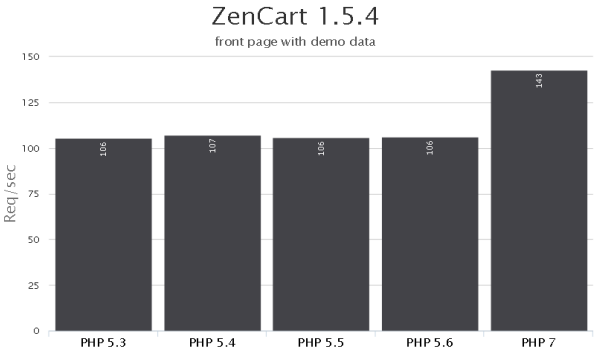


*Figura 4 – Request/segundo que realiza el CMS ZenCart dependiendo de la versión de PHP*

## 4. Conclusiones

Que más decir, los creadores de PHP, sobretodo Rasmus Lerdorf, no ignoraron las críticas hacia su lenguaje, dan un salto significativo con todas las características que trae, siendo las más impresionantes:

- Mejora significativa de la API
- Funciones de retorno de datos
- Operador de fusión null
- Operador Spaceship o RFC

En fin desde sus versiones 3, 4 y 5 los cambios en php han sido tan poco susceptibles, en cuanto a eficiencia, que daba lo mismo programar en PHP 4 o en PHP 5.6; pero con su nueva versión, esta vez nos ofrece más orden y elegancia en el código para nosotros como programadores y más eficiencia en el producto terminado para los usuarios finales.

**Referencias bibliográficas:**


1 – php.net, Página oficial "Historia de PHP", citado de internet, del url: http://php.net/manual/es/history.php.php , el 03 octubre de 2015.

2 - Ley, Leroy (26 septiembre, 2015) "PHP 7: Lo nuevo", citado de internet, del url: http://mycyberacademy.com/php-7-lo-nuevo/, el 03 de octubre de 2015.

3 - Lerdorf, Rasmus (25 Abril, 2015) "SPEEDING UP THE WEB WITH PHP 7", citado de internet, del url: http://talks.php.net/fluent15, el 03 de octubre de 2015.

4 - php.net, Página oficial "PHP RFC: Combined Comparison (Spaceship) Operator", citado de internet, del url: https://wiki.php.net/rfc/combined-comparison-operator, el 03 de octubre de 2015.

5 - php.net, Página oficial "Migrating from PHP 5.6.x to PHP 7.0.x", citado de internet, del url: http://php.net/manual/en/migration70.php, el 05 octubre de 2015.

6 - Lerdorf, Rasmus "Documentation for the php7dev Vagrant box image", citado de internet, del url: https://github.com/rlerdorf/php7dev, el 05 de octubre de 2015.