

# Lenguajes de Script

En el servidor web: PHP

## INDICE DE CONTENIDOS

### 1) Iniciación a PHP 7

- 1.1) Introducción a PHP 7
- 1.2) Instalación de Software necesario para PHP 7
- 1.3) Nuestro primer PHP ¡Hola mundo! y algo más
- 1.4) Comentarios dentro de código PHP 7
- 1.5) Variables en PHP 7
- 1.6) Variables reservadas en PHP 7
- 1.7) Cómo concatenar en PHP 7

### 2) Operadores en PHP 7

- 2.1) Operadores aritméticos en PHP 7
- 2.2) Operadores de comparación PHP 7
- 2.3) Operadores lógicos en PHP 7

### 3) Instrucciones en PHP 7

- 3.1) Condicionales en PHP 7
- 3.2) Qué son los bucles PHP 7
- 3.3) Salida función printf PHP 7
- 3.4) Manejo de cadenas PHP 7

### 4) Funciones en PHP 7

- 4.1) ¿Qué son las funciones en PHP 7?

### 5) Procesado de formularios con PHP 7

- 5.1) Recuperar datos en métodos GET y POST
- 5.2) ¿Qué es la función mail en PHP 7?

### 6) Bases de Datos MySQL en PHP 7

- 6.1) Crear la base de datos y tabla MySQL en PHP 7
- 6.2) Conectarse a la Base de Datos de MySQL en PHP 7
- 6.3) Consultas a la Base de Datos MySQL en PHP 7
- 6.4) Insertar registros a la Base de Datos MySQL en PHP 7
- 6.5) Actualizar registros de Base de Datos MySQL en PHP 7
- 6.6) Borrar registros de la Base de Datos MySQL en PHP 7

### 7) Sesiones en PHP 7

- 7.1) ¿Qué son las sesiones en PHP 7?
- 7.2) Ejemplo de uso de sesiones en PHP 7

### 8) Cookies en PHP 7

- 8.1) ¿Qué son las Cookies en PHP 7?
- 8.2) Ejemplos de cómo usar Cookies en PHP 7

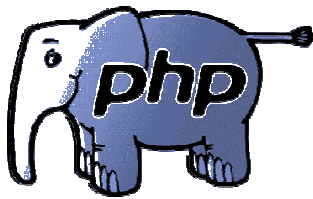
## **9) Expresiones regulares EE.RR en PHP 7**

- 9.1) ¿Qué son las expresiones regulares en PHP 7?
- 9.2) Veamos 10 ejemplos sobre patrones de expresiones regulares
- 9.3) Ejemplos de expresiones regulares

## **10) PHP 7 Orientado a Objetos**

- 10.1) ¿Qué son las clases en PHP 7?
- 10.2) Los atributos de POO en PHP 7
- 10.3) ¿Qué es la Herencia POO en PHP 7?
- 10.4) Veamos el Acceso Public (Público) en POO PHP 7
- 10.5) Veamos el Acceso Private (Privado) en POO PHP 7

## Capítulo 1.- Introducción a PHP 7



*Este es el primer capítulo de nuestro **Tutorial PHP 7**, veremos una breve intruducción a PHP 7, vamos a instalar el software necesario para su ejecución, veremos los comentarios dentro del código, las variables y como concatenar.*

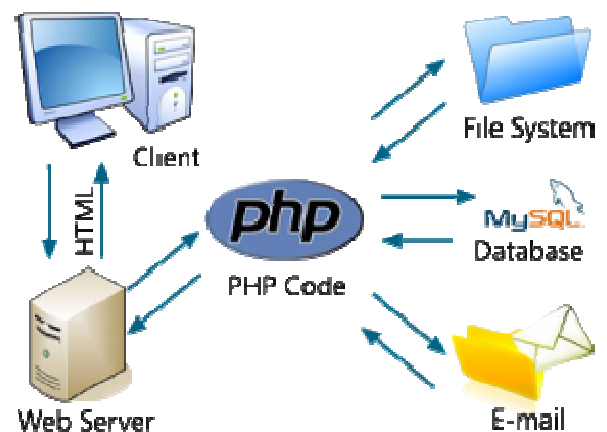
## 1.1) Introducción a PHP 7

PHP es un lenguaje de programación creado en el año 1995 de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos.

El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante. PHP ha evolucionado por lo que ahora incluye también una interfaz de línea de comandos que puede ser usada en aplicaciones gráficas independientes. Puede ser usado en la mayoría de los servidores web al igual que en casi todos los sistemas operativos y plataformas sin ningún costo.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocidos hasta el día de hoy, lo que ha atraído el interés de múltiples sitios con gran demanda de tráfico, como Facebook, para optar por el mismo como tecnología de servidor.

Con las primeras 2 versiones de PHP, PHP 3 y PHP 4, se había conseguido una plataforma potente y estable para la programación de páginas del lado del servidor. Estas versiones han servido de mucha ayuda para la comunidad de desarrolladores, haciendo posible que PHP sea el lenguaje más utilizado en la web para la realización de páginas avanzadas.



Sin embargo, todavía existían puntos negros en el desarrollo PHP que se han tratado de solucionar con la versión 5, aspectos que se echaron en falta en la versión 4, casi desde el día de su lanzamiento. Nos referimos principalmente a la programación orientada a objetos (POO) que, a pesar de que estaba soportada a partir de PHP 3, sólo implementaba una parte muy pequeña de las características de este tipo de programación.

**Nota:** La orientación a objetos es una manera de programar que trata de modelar los procesos de programación de una manera cercana a la realidad: tratando a cada componente de un programa como un objeto con sus características y funcionalidades.

El principal objetivo de PHP 7 ha sido mejorar los mecanismos de POO para solucionar las carencias de las anteriores versiones. Un paso necesario para conseguir que PHP sea un lenguaje apto para todo tipo de aplicaciones y entornos, incluso los más exigentes.

## 1.2) Instalación de Software necesario para PHP 7



Como ya hemos leído, **PHP** es un lenguaje de programación de **lado servidor**, lo que significa que todo el código **PHP** será interpretado por el **Binario PHP** en el Servidor Web, y el resultado será enviado al cliente, comunmente puede ser:

- HTML
- JavaScript
- CSS
- Imágenes
- Cualquier texto sin formato

Y éste, es entregado por el servidor **HTTP web**.

Para la realización del tutorial recomendamos instalar un servidor en nuestra misma máquina, a modo de pruebas. También puedes hacerlo en tu hosting (si ya lo tienes, aunque para esta altura aún no vale la pena contratarlo). Para nuestro Tutorial PHP 7 usaremos **XAMPP**.



### ¿Qué es XAMPP?

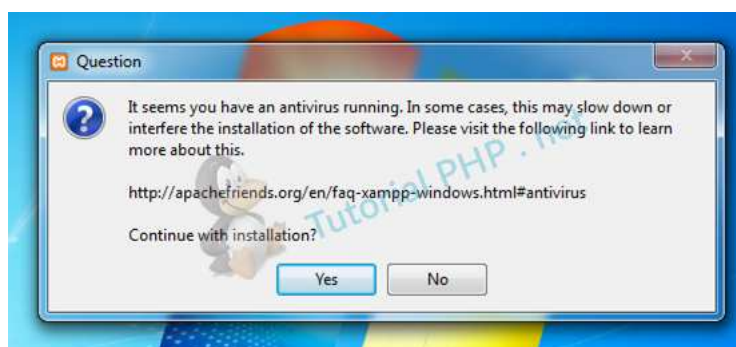
**XAMPP** es una distribución de Apache completamente gratuita y fácil de instalar que contiene el **servidor web HTTP Apache**, **Base de Datos MySQL**, **PHP** y **Perl**. El paquete de instalación de **XAMPP** ha sido diseñado para ser increíblemente fácil de instalar y usar, es muy potente, tanto para poder desarrollar, probar, hasta para montar un servidor de producción, solo que éste último no lo recomendamos (en lo personal).

A continuación instalaremos **XAAMP versión 5.6.8 para Windows 7 32 bits**.

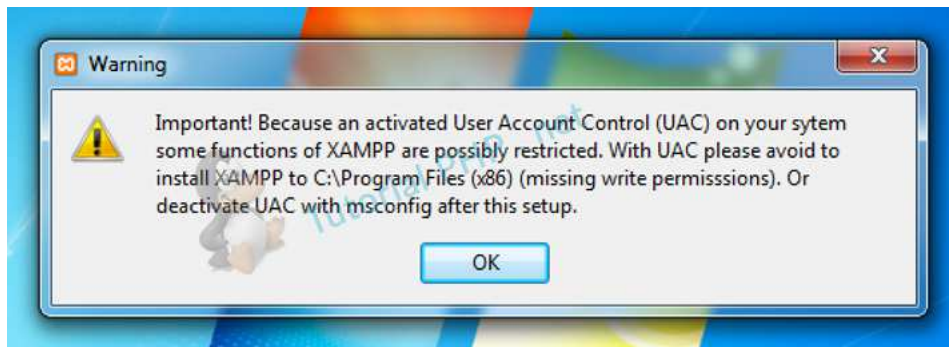
La liga para su descarga está aquí:

<https://www.apachefriends.org/xampp-files/5.6.8/xampp-win32-5.6.8-0-VC11-installer.exe>

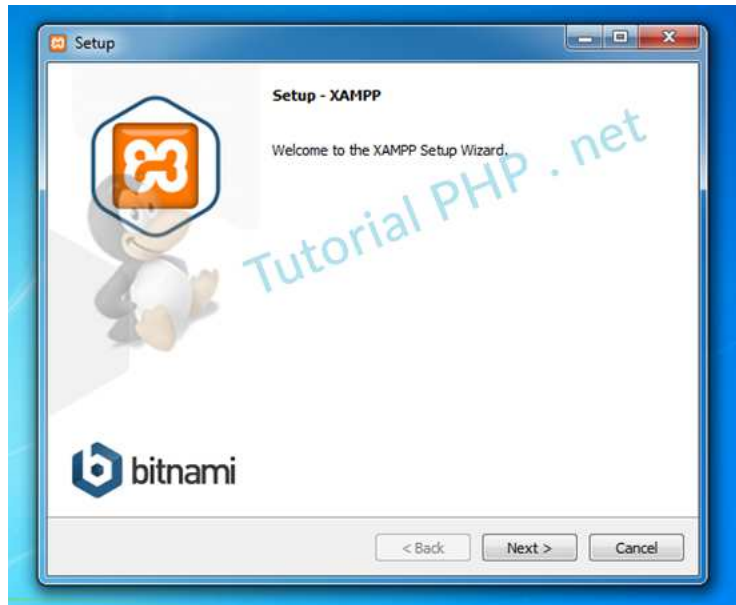
Una vez descargado, ejecutaremos el instalador **xampp-win32-5.6.8-0-VC11-installer.exe**.



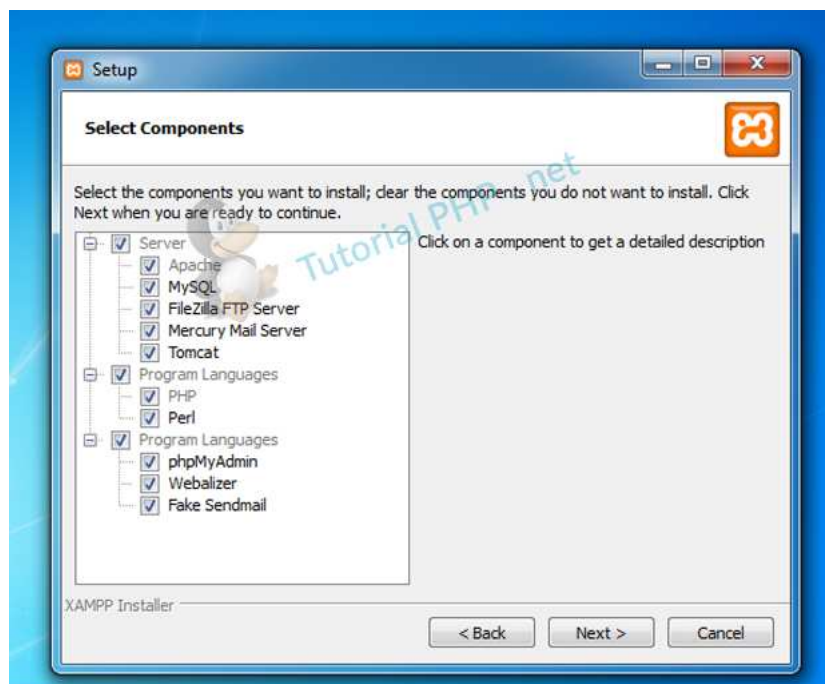
Clickeamos en **"YES"** y continuamos.



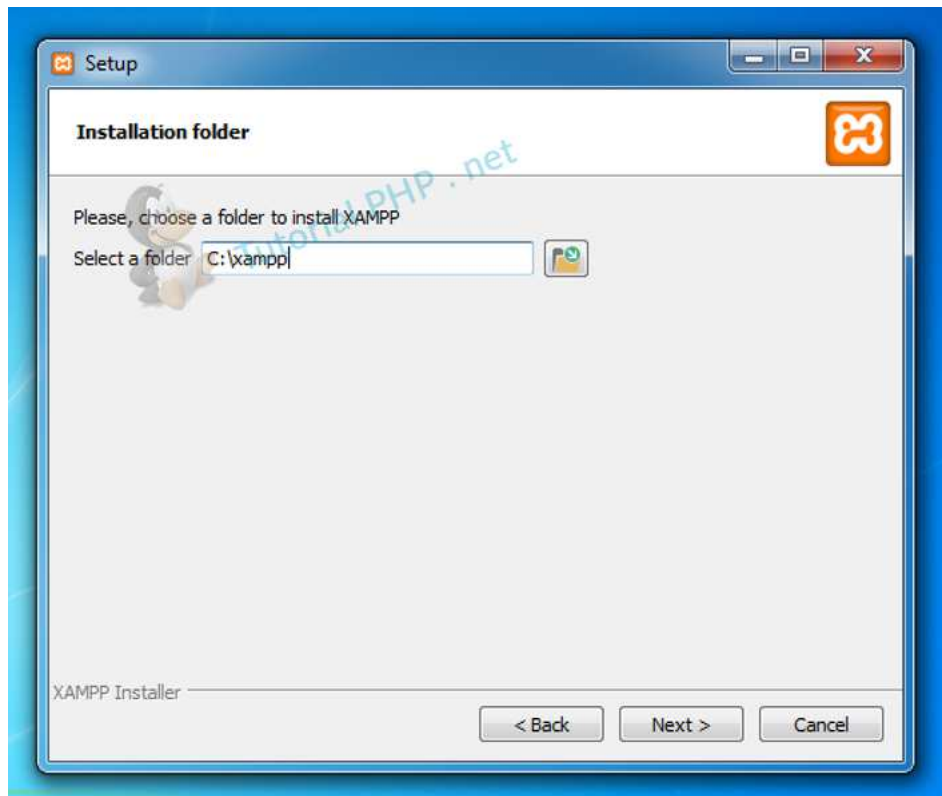
Clickeamos en "OK" y continuamos.



Clickeamos en "Next >" y continuamos.



Lo dejamos tal cual, clickeamos "Next >" y continuamos.

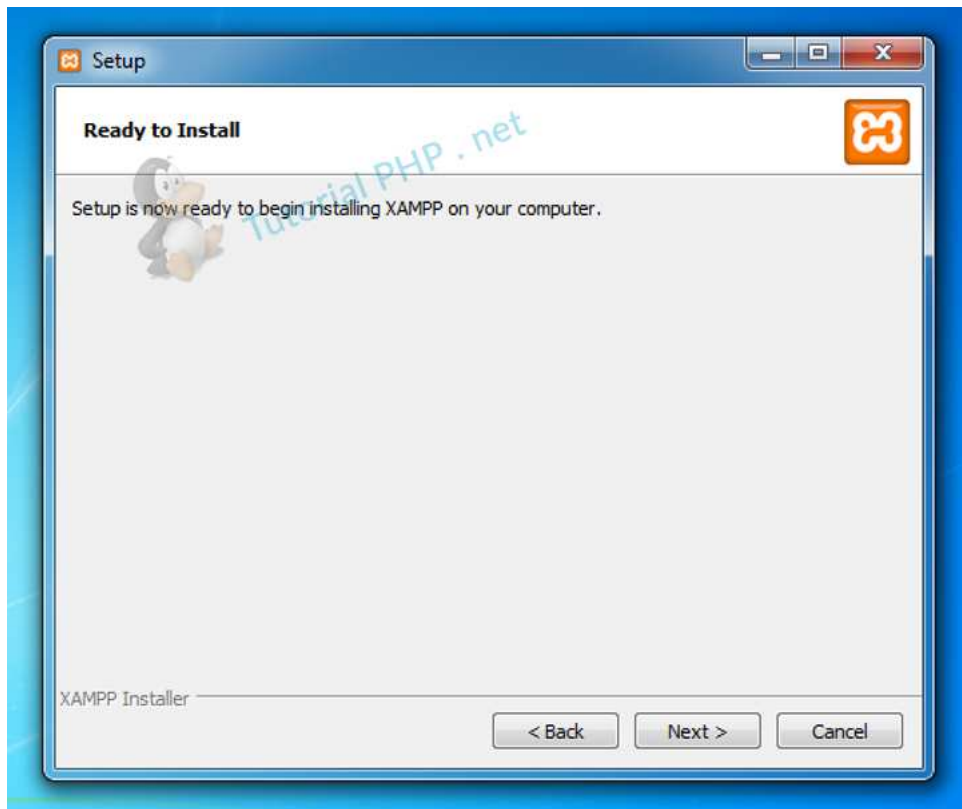


Nos preguntara por la ruta de instalación que queremos darle, dejaremos la default que viene (así como está en pantalla), clickeamos **"Next >"** y continuamos.



Clickeamos en **"Next >"** y continuamos.





Clickeamos en "**Next >**" y continuamos.

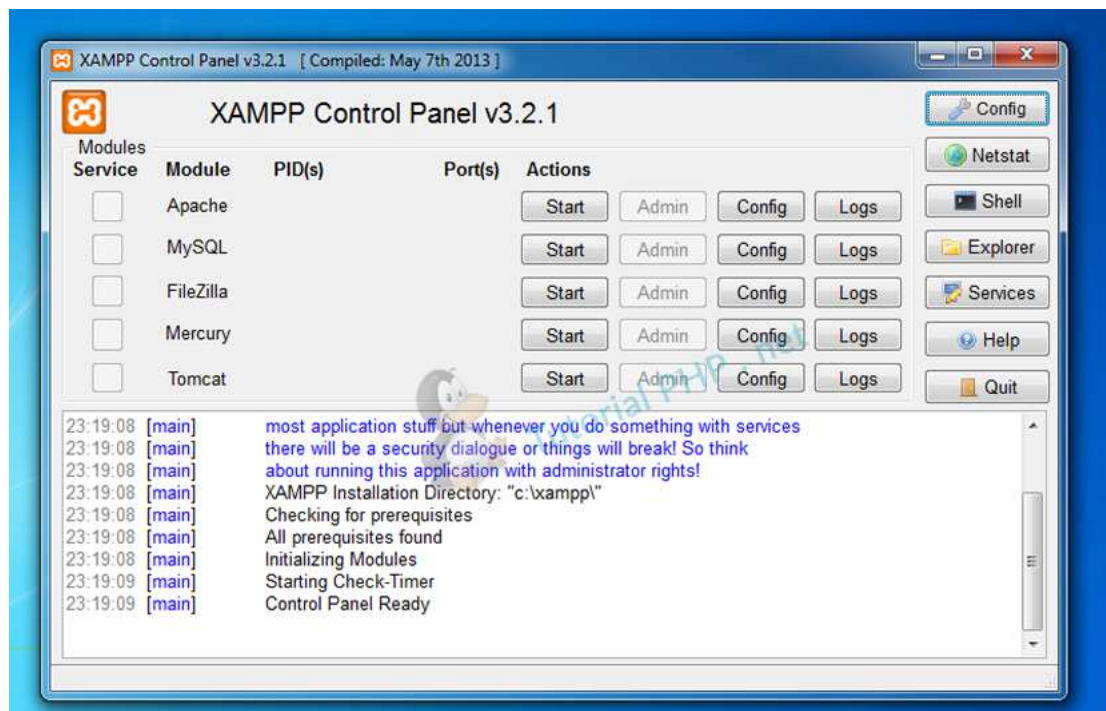


Comenzará a instalar, esperamos.

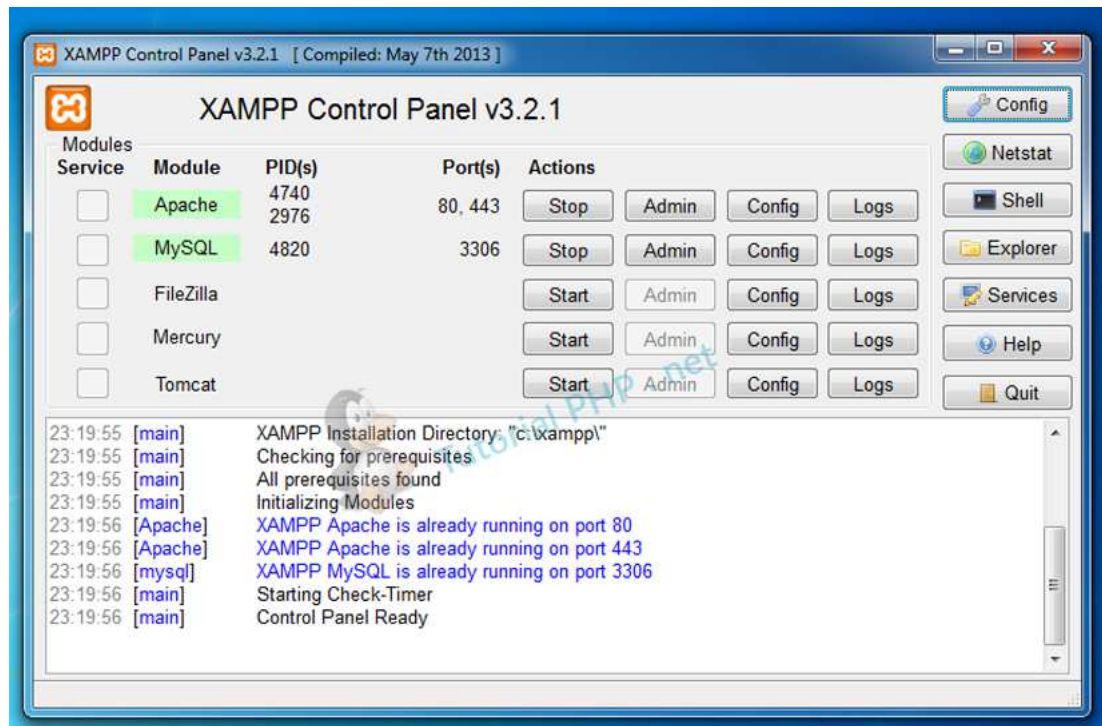


Una vez finalizado, nos preguntará si deseamos abrir el **Panel de Control**, seleccionamos la casilla y clickeamos en "**Finish**".

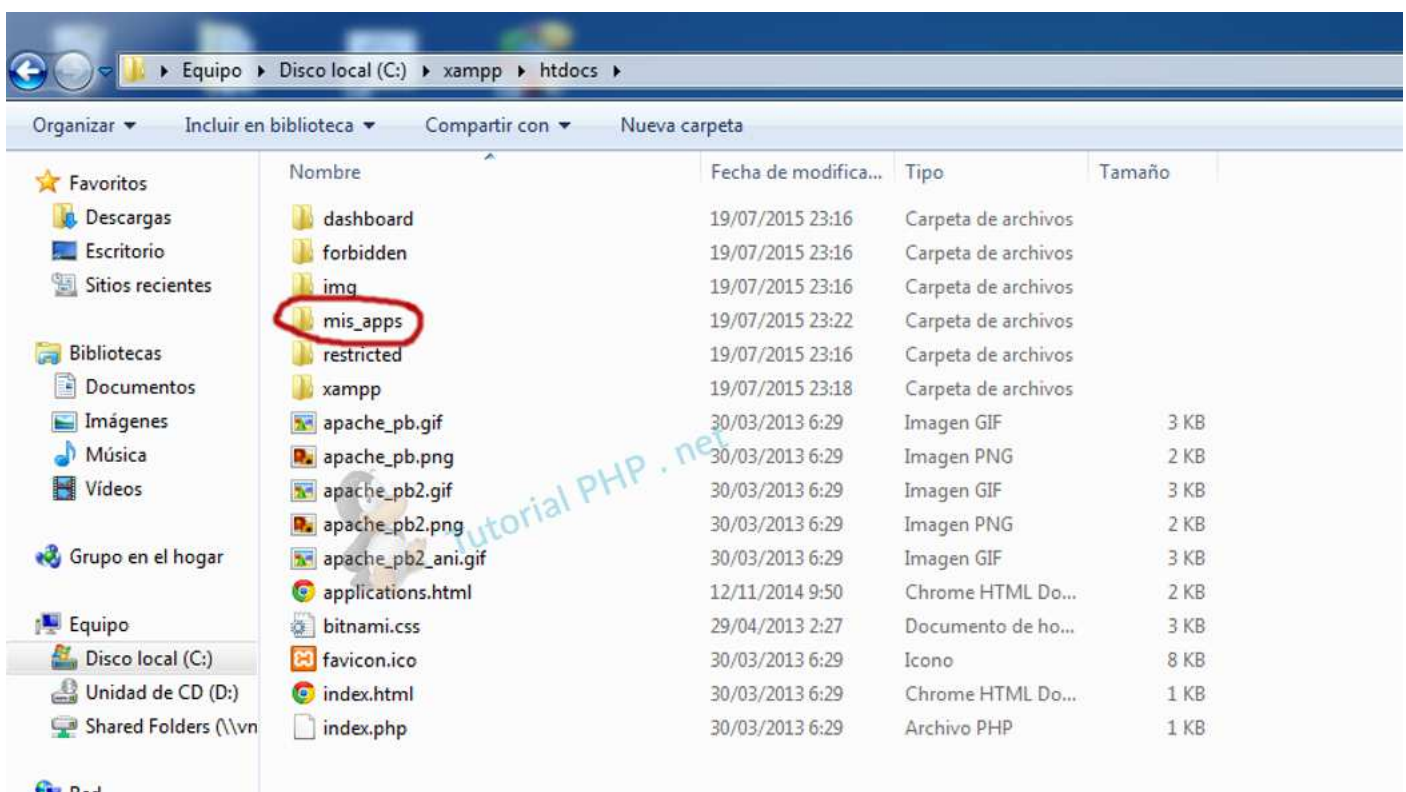
Listo. Ya tenemos nuestro paquete de servidores locales instalados. Ahora, se abrirá el **Panel de Control**.



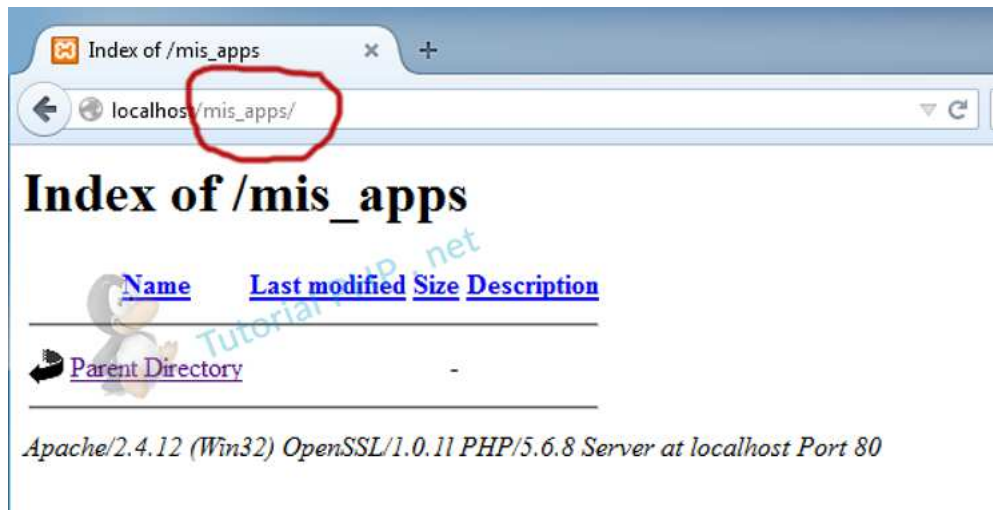
En el **Panel de Control**, tenemos la opción de iniciar, reiniciar o parar el servidor que deseemos, así como los accesos directos a los archivos de configuración (éstos por el momento no los tocaremos).



Arrancamos los servidores **Apache (web HTTP)** y de **Base de datos MySQL**.



Nuestra carpeta raíz donde se alojarán todos los **archivos PHP** está en "**C:\xampp\htdocs\**" y yo creé una nueva carpeta llamada "**mis\_apps**", creenla.



Abriremos nuestro navegador, e iremos a la dirección [http://localhost/mis\\_apps/](http://localhost/mis_apps/). Aquí iremos poniendo nuestros **archivos PHP** generados en este **Tutorial de PHP 7**.

Para acceder a las Bases de Datos MySQL lo puedes hacer desde <http://localhost/phpmyadmin/>.

Felicidades, ya tenemos nuestro ambiente de desarrollo listo.

### 1.3) Nuestro primer PHP ¡Hola mundo! y algo más

En este capítulo aprenderemos a crear un archivo **PHP** que nos devuelva un código dinámico. Ejemplo, un "**Hola mundo**" en conjunto a saber intercalar el código PHP y otros códigos (HTML, JavaScript, CSS, etc).

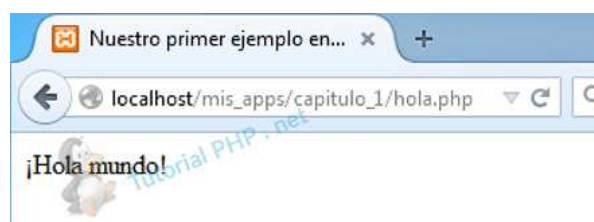
Para ello, crearemos una carpeta dentro de "**C:\xampp\htdocs\mis\_apps\**" que se llame "**capitulo\_1**" y dentro de éste, un archivo llamado **hola.php**. A continuación escribiremos lo siguiente:

```
<?php

echo "¡Hola mundo!";

?>
```

Vamos a abrir nuestro explorador y entraremos en localhost/mis\_apps/capitulo\_1/hola.php. Deberemos ver lo siguiente:



Siempre que vayamos a escribir código PHP 7 debemos comenzar con "**<?php**" y terminar con "**?>**", **echo** es una función PHP propia para imprimir en pantalla lo que siga entre " ". Si todo resultó bien, ¡felicidades!. Creaste **tu primer código PHP 7 dinámico**.

La ventaja que tiene **PHP 7** sobre otros lenguajes de programación que se ejecutan en el servidor (*como podrían ser los script CGI Perl*), es que nos permite intercalar las sentencias **PHP 7** en las páginas HTML, es un concepto algo complicado de entender si no se ha visto nunca como funciona unas páginas PHP, ASP, .NET, etc.

Vamos a ver el siguiente ejemplo para comprenderlo mejor. En otro archivo, llamado **hola\_bucle.php** escribiremos lo siguiente:

```
<html>
<head>
  <title>Nuestro primer ejemplo en PHP 7</title>
</head>

<body>

  <?php

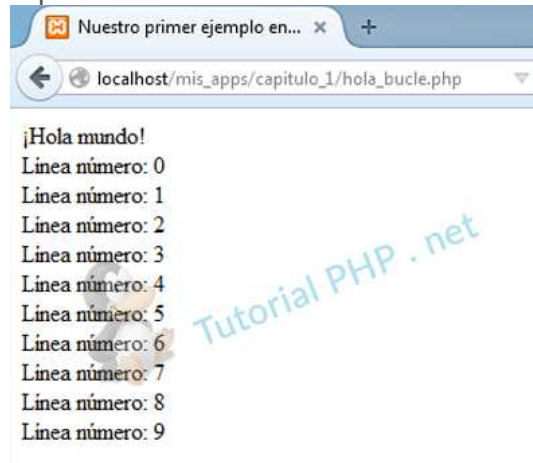
  echo "¡Hola mundo! <br>";

  for($i = 0; $i < 10; $i++)
  {
    echo "Línea número: ".$i."<br>";
  }

  ?>

</body>
</html>
```

El código **PHP** ejecutado tiene dos partes: la primera imprime "**¡Hola mundo!**" y la segunda es un bucle que se ejecuta 10 veces de 0 a 9, por cada vez que se ejecuta se escribe una línea, la variable **\$i** contiene el número de línea que se está escribiendo.



No importa si no entiende muy bien el programa este ejemplo solo es para ilustrar como se intercala el código HTML y el código PHP.

## 1.4) Comentarios dentro de código PHP 7

En PHP 7 como en todos los lenguajes de programación y de marcado (HTML, XML, CSS, etc) existen los comentarios. Pero, ¿qué son los comentarios?, bueno los comentarios son fragmentos de texto dentro de código de programación que a través de algunos caracteres al inicio, el **binario de PHP** los ignorará, en PHP hay tres tipos:

- 1.- De doble barra //
- 2.- Con signo de gato #
- 3.- Con barra y asterisco para apertura y asterisco y barra para cierre /\* \*/

Ahora veamos los ejemplos:

### 1.- De doble barra //

```
<html>
<head>
  <title>Ejemplo de comentarios en PHP 7</title>
</head>

<body>

<?php

// ¡Este es el primer tipo de comentario!. Podemos agregar tanto texto como queramos sin cambiar de renglón.

echo "Dejaremos este echo como ejemplo de ejecución de PHP 7";

?>

</body>
</html>
```

### 2.- Con signo de gato #

```
<html>
<head>
  <title>Ejemplo de comentarios en PHP 7</title>
</head>

<body>

<?php

# ¡Este es el segundo tipo de comentario!. Podemos agregar tanto texto como querramos sin cambiar de renglón.

echo "Dejaremos este echo como ejemplo de ejecución de PHP 7";

?>

</body>
</html>
```

### 3.- Con barra y asterisco para apertura y asterisco y barra para cierre /\* \*/

```
<html>
<head>
```

```
<title>Ejemplo de comentarios en PHP 7</title>
</head>

<body>

<?php

/*
 ¡Este es el segundo tipo de comentario!.
 Podemos agregar tanto texto como queramos cambiando de renglón.
 Sin importar que escribamos
 Bajando de renglones
 Hasta su cierre
 */

echo "Dejaremos este echo como ejemplo de ejecución de PHP 7";

?>

</body>
</html>
```

Usar comentarios dentro de código PHP 7 nos es muy útil ya que podemos dejar descripciones o lo que deseemos.



## 1.5) Variables en PHP 7

Las variables son uno de los primeros temas que tenemos que conocer en PHP 7 y en la mayoría de los lenguajes de programación.

Una variable es un contenedor de información, en el que podemos almacenar números enteros, números decimales, cadenas de texto, etc. El contenido de las variables se puede leer y se puede cambiar durante la ejecución de una página que tenga PHP 7.

En PHP todas las variables comienzan con el símbolo del dólar \$ y no es necesario definir una variable antes de usarla.

Tampoco tienen tipos, es decir que una misma variable puede contener un número y luego puede contener caracteres.

**OJO:** Son sensibles a minúsculas y mayúsculas y tampoco se deben de iniciar con un número (más abajo especificamos ejemplos inválidos). Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_1**" crearemos un archivo llamado **variables.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de tipos de variables en PHP 7</title>
</head>

<body>

<?php

$a = 1;

$b = "3.34";

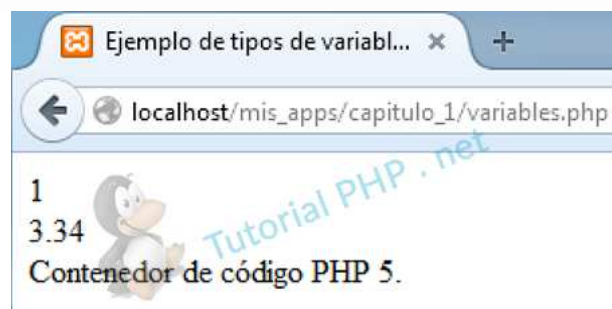
$c = "Contenedor de código PHP 7.";

// Veamos qué viene

echo $a."<br>".$b."<br>".$c;

?>

</body>
</html>
```



Un punto importante a resaltar, es que si vamos a utilizar una variable que está especificada afuera de alguna función que desarrollemos tenemos que anteponer **global**. Veamos un ejemplo, crearemos un archivo llamado **variables\_funcion.php**:

```
<html>
```

```
<head>
  <title>Ejemplo de tipos de variables en PHP 7</title>
</head>

<body>

  <?php

$a = 1;

$b = "3.34";

$c = "Contenedor de código PHP 7 en una función.";

function ejemploVariables()
{
  global $a, $b, $c;

  // Veamos qué viene

  echo $a."<br>".$b."<br>".$c;

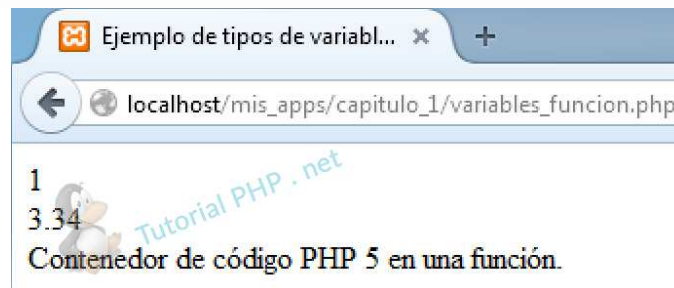
}

// Ejecutamos la función

ejemploVariables();

?>

</body>
</html>
```



También, podemos almacenar información en variables mediante matrices (las matrices las veremos más adelante), veamos un ejemplo, crearemos un archivo llamado **variables\_array.php**:

```
<html>
<head>
  <title>Ejemplo de tipos de variables en PHP 7</title>
</head>

<body>

  <?php

  // En primer lugar indicamos que $nuestraVariable es una matriz

  $nuestraVariable = array();

  // Asignemos valores
```

```

$nuestraVariable[0] = "Tenemos la posición en 0";
$nuestraVariable[1] = "Tenemos la posición en 1";
$nuestraVariable[2] = "Tenemos la posición en 2";
$nuestraVariable[3] = "Tenemos la posición en 3";
$nuestraVariable[4] = "Tenemos la posición en 4";

// Recorramos la matriz

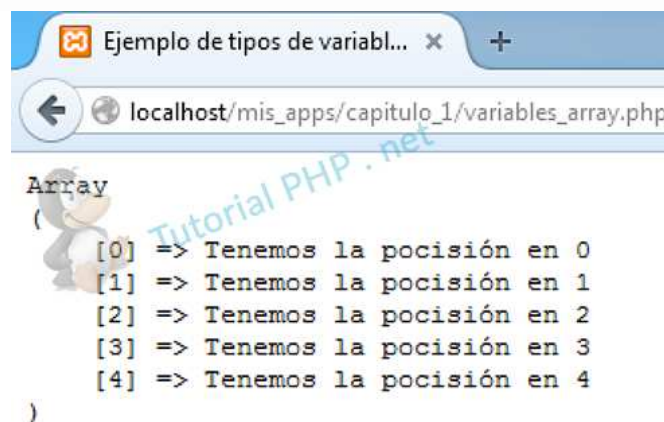
echo "<pre>"; // <- HTML que especifica texto sin formato

print_r($nuestraVariable);

echo "</pre >";
?>

</body>
</html>

```



Con el ejemplo anterior, vemos que también podemos asignarle valores a una variable matriz.

A continuación ejemplos inválidos:

Possible nombre de variable	¿Es válido?
<b>\$4variable</b>	Nombre de variable <b>inválido</b> .
<b>\$_4variable</b>	Nombre de variable válido.
<b>\$variable4</b>	Nombre de variable válido.
<b>\$otra</b>	Nombre de variable válido.
<b>\$1_otra</b>	Nombre de variable <b>inválido</b> .
<b>\$variable_de_nombre_muy_largo</b>	Nombre de variable válido.
<b>\$ABC</b>	Nombre de variable válido.
<b>\$ A B C</b>	Nombre de variable <b>inválido</b> .
<b>\$A_y_B_x_C</b>	Nombre de variable válido.

## 1.6) Variables reservadas en PHP 7

Las variables reservadas son aquellas que PHP 7 define con valores en matrices por default, que están disponibles todo el tiempo sin necesidad de requerir librerías y que no podemos cambiar su contenido. A continuación veremos una tabla descriptiva y después, ya en detalle cada una:

Variable	Valor
\$_SERVER	Información del entorno del servidor y de ejecución.
\$_GET	Variables en el encabezado HTTP GET
\$_POST	Variables recibidas en el encabezado HTTP GET
\$_COOKIE	Variable con la cual podemos crear, acceder, editar o destruir Cookies.
\$_FILES	Variables que llegan al servidor con archivos mediante carga.
\$_REQUEST	Es una variable de array asociativo que por defecto contiene el contenido de \$_GET, \$_POST y \$_COOKIE.
\$_SESSION	Variables de sesión.

### \$\_SERVER

La variable **\$\_SERVER** nos devolverá en forma de array (matriz) información de servidor, rutas, conexiones, información del cliente y distintos *headers* recibidos. Veamos un ejemplo, dentro de nuestra carpeta "mis\_apps\capitulo\_1" crearemos un archivo llamado **variables\_server.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de variable reservada en PHP 7</title>
</head>

<body>

<pre>

<?php

print_r($_SERVER);

?>

</pre >

</body>
</html>
```

```

Ejemplo de variable reserva... x
localhost/mis_apps/capitulo_1/varibales_server.php

Array
(
    [MIBDIRS] => C:/xampp/php/extras/mibs
    [MYSQL_HOME] => \xampp\mysql\bin
    [OPENSSL_CONF] => C:/xampp/apache/bin/openssl.cnf
    [PHP_FEAR_SYSCONF_DIR] => \xampp\php
    [PHPRC] => \xampp\php
    [TMP] => \xampp\tmp
    [HTTP_HOST] => localhost
    [HTTP_USER_AGENT] => Mozilla/5.0 (Windows NT 6.1; WOW64; rv:39.0) Gecko/20100101 Firefox/39.0
    [HTTP_ACCEPT] => text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
    [HTTP_ACCEPT_LANGUAGE] => es-MX,es-ES;q=0.9,es;q=0.7,es-AR;q=0.6,es-CL;q=0.4,en-US;q=0.3,en;q=0.1
    [HTTP_ACCEPT_ENCODING] => gzip, deflate
    [HTTP_CONNECTION] => keep-alive
    [PATH] => C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\
    [SystemRoot] => C:\Windows
    [COMSPEC] => C:\Windows\system32\cmd.exe
    [PATHEXT] => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
    [WINDIR] => C:\Windows
    [SERVER_SIGNATURE] =>

Apache/2.4.12 (Win32) OpenSSL/1.0.11 PHP/5.6.8 Server at localhost Port 80

    [SERVER_SOFTWARE] => Apache/2.4.12 (Win32) OpenSSL/1.0.11 PHP/5.6.8
    [SERVER_NAME] => localhost
    [SERVER_ADDR] => ::1
    [SERVER_PORT] => 80
    [REMOTE_ADDR] => ::1
    [DOCUMENT_ROOT] => C:/xampp/htdocs
    [REQUEST_SCHEME] => http
    [CONTEXT_PREFIX] =>
    [CONTEXT_DOCUMENT_ROOT] => C:/xampp/htdocs
    [SERVER_ADMIN] => postmaster@localhost
    [SCRIPT_FILENAME] => C:/xampp/htdocs/mis_apps/capitulo_1/varibales_server.php
    [REMOTE_PORT] => 49504
    [GATEWAY_INTERFACE] => CGI/1.1
    [SERVER_PROTOCOL] => HTTP/1.1
    [REQUEST_METHOD] => GET
    [QUERY_STRING] =>
    [REQUEST_URI] => /mis_apps/capitulo_1/varibales_server.php
    [SCRIPT_NAME] => /mis_apps/capitulo_1/varibales_server.php
    [PHP_SELF] => /mis_apps/capitulo_1/varibales_server.php
    [REQUEST_TIME_FLOAT] => 1438834878.384
    [REQUEST_TIME] => 1438834878
)

```

## \$\_GET

La variable **\$\_GET** nos devolverá en forma de array (matriz) información de variables enviadas a través del parámetro HTTP GET, es decir, en la dirección de solicitud, variables y asignaciones con **&** = (*archivo.php?variable-1=valor1&varibale-2=valor2&varibale-3=valor3*).

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_1**" crearemos un archivo llamado **variables\_get.php** con el siguiente código y lo ejecutaremos con los siguientes parametros:

[http://localhost/mis\\_apps/capitulo\\_1/variables\\_get.php?variable-1=valor1&varibale-2=valor2&varibale-3=valor3](http://localhost/mis_apps/capitulo_1/variables_get.php?variable-1=valor1&varibale-2=valor2&varibale-3=valor3)

```
<html>
<head>
  <title>Ejemplo de variable reservada en PHP 7</title>
</head>

<body>

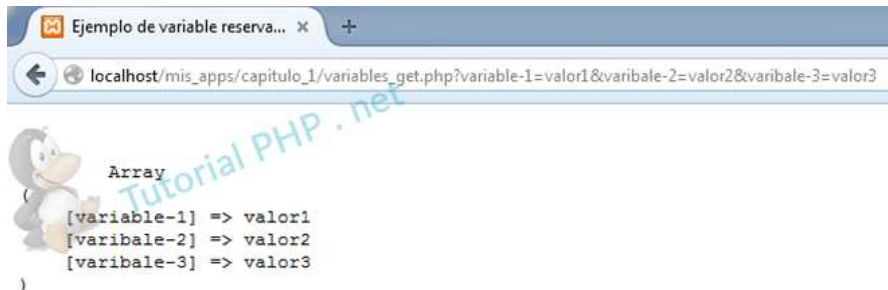
<pre>

<?php
print_r($_GET);

?>

</pre >

</body>
</html>
```



## \$\_POST

La variable **\$\_POST** nos devolverá en forma de array (matriz) información de variables enviadas a través del paramatro HTTP POST. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_1**" crearemos un archivo llamado **variables\_post.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de variable reservada en PHP 7</title>
</head>

<body>

<h2>Ejemplo de variables con protocolo POST</h2>

<form method="post">
```

```

Escribe tu nombre: <input type="text" name="nombre" value="<?=@$_POST['nombre'];?>"> <br>
<br>

Escribe tu edad: <input type="text" name="edad" value="<?=@$_POST['edad'];?>"> <br> <br>

<input type="submit" value="Enviar">

</form>

<pre>

<?php

if($_POST)
{
    print_r($_POST);
}

?>

</pre >

</body>
</html>

```



## \$\_COOKIE

La variable **\$\_COOKIE** nos devolverá en forma de array (matriz) información de cookies guardadas en nuestro cliente. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_1**" crearemos un archivo llamado **variables\_cookie.php** con el siguiente código:

```

<?php

// Establecemos los valores de las Cookies

setcookie("Valor_1", "1");

setcookie("Valor_2", "2");

setcookie("Valor_3", "3");

```

```
?>

<html>

<head>
  <title>Ejemplo de variable reservada en PHP 7</title>
</head>

<body>

<pre>

<?php

    print_r($_COOKIE);

?>

</pre >

</body>
</html>
```



## \$\_FILES

La variable **\$\_FILES** nos devolverá en forma de array (matriz) información de archivos que hayan llegado al servidor a través del protocolo HTTP POST. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_1**" crearemos un archivo llamado **variables\_file.php** con el siguiente código:

```
<html>

<head>
  <title>Ejemplo de variable reservada en PHP 7</title>
</head>

<body>

<h2>Ejemplo de variables FILES POST</h2>

<form method="post" enctype="multipart/form-data">

  Archivo: <input type="file" name="archivo"> <br> <br>

<input type="submit" value="Enviar">

</form>
```



```
<pre>

<?php

if($_FILES)
{
    print_r($_FILES);
}

?>

</pre>

</body>
</html>
```



## \$\_REQUEST

La variable **\$\_REQUEST** nos devolverá en forma de array (matriz) información en array asociativo con el contenido de las variables **\$\_GET**, **\$\_POST** y **\$\_COOKIE**. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_1**" crearemos un archivo llamado **variables\_request.php** con el siguiente código:

```
<html>

<head>
    <title>Ejemplo de variable reservada en PHP 7</title>
</head>

<body>

<pre>

<?php

    print_r($_REQUEST);

?>

</pre>

</body>
```

```
</html>
```

## **\$\_SESSION**

La variable **\$\_SESSION** nos devolverá en forma de array (matriz) información de variables establecidas en sesión, estando accesibles en cualquier parte de nuestra aplicación sin necesidad de requerir las definiciones. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_1**" crearemos un archivo llamado **variables\_session.php** con el siguiente código:

```
<?php
```

```
// Siempre que utilicemos sesiones deberemos iniciar con session_start.
```

```
session_start();
```

```
$_SESSION['variable_de_sesion_1'] = "Algún valor definido";
```

```
$_SESSION['variable_de_sesion_2'] = "Algún otro valor definido";
```

```
?>
```

```
<html>
```

```
<head>
```

```
<title>Ejemplo de variable reservada en PHP 7</title>
```

```
</head>
```

```
<body>
```

```
<pre>
```

```
<?php
```

```
print_r($_SESSION);
```

```
?>
```

```
</pre >
```

```
</body>
```

```
</html>
```

## 1.7) Cómo concatenar en PHP 7

En PHP 7 al igual que en muchos lenguajes de programación existe la concatenación, pero ¿qué es?. La concatenación es pegar información que venga en una variable o en una función. Veamos un ejemplo, dentro de nuestra carpeta "mis\_apps\capitulo\_1" crearemos un archivo llamado **concatenacion.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de concatenación en PHP 7</title>
</head>

<body>

  <?php

$nombre = "Pedro Martínez";

$ciudad = "Bogotá";

$pais = "Colombia";

$edad = "30";

echo "Hola, mi nombre es ".$nombre.", vivo en la ciudad de ".$ciudad.", en el país ".$pais." y
tengo ".$edad." años.";

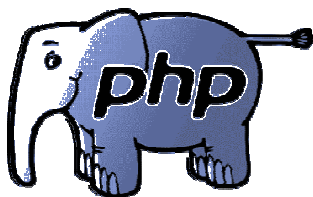
?>

</body>
</html>
```



Hola, mi nombre es Pedro Martínez, vivo en la ciudad de Bogotá, en el país Colombia y tengo 30 años.

## Capítulo 2.- Operadores en PHP 7



En el segundo capítulo de nuestro Tutorial PHP 7 veremos los operadores aritméticos, de comparación y lógicos.

## 2.1) Operadores aritméticos en PHP 7

Los operadores de **PHP 7** son muy parecidos a los lenguajes de programación como C, C++, *JavaScript* o *Python*, si conoces estos lenguajes te resultaran muy familiares y fáciles de reconocer. Si no, no te preocupes, vamos a aprenderlos.

Estos son los operadores que se pueden aplicar a las variables y constantes numéricas.

Operador	Nombre	Ejemplo	Descripción
+	Suma	5 + 6	Suma dos números
−	Resta	7 − 9	Resta dos números
*	Multiplicación	6 * 3	Multiplica dos números
/	División	4 / 9	Divide dos números
%	Módulo	7 % 2	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1
++	Suma 1	\$a++	Suma 1 al contenido de una variable
—	Resta 1	\$a−	Resta 1 al contenido de una variable

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_2**" crearemos un archivo llamado **operadores\_aritmeticos.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de operadores aritméticos en PHP 7</title>
</head>

<body>

<?php

$a = 8;

$b = 3;

echo $a + $b."<br>";

echo $a - $b."<br>";

echo $a * $b."<br>";

echo $a / $b."<br>";

echo $a % $b."<br>";

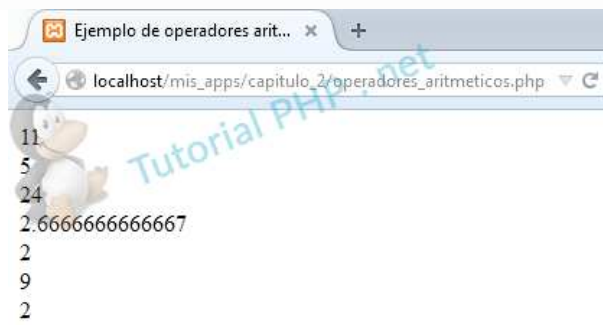
$a++;
echo $a."<br>";

$b--;
echo $b."<br>";

?>

</body>
```

```
</html>
```



## 2.2) Operadores de comparación PHP 7

Los operadores de comparación, como su nombre lo indica, permiten comparar dos valores. Veámos la siguiente tabla:

Operador	Nombre	Ejemplo	Devuelve cierto cuando
==	Igual	\$a == \$b	\$a es igual \$b
!=	Distinto	\$a != \$b	\$a es distinto \$b
<	Menor que	\$a < \$b	\$a es menor que \$b
>	Mayor que	\$a > \$b	\$a es mayor que \$b
<=	Menor o igual	\$a <= \$b	\$a es menor o igual que \$b
>=	Mayor o igual	\$a >= \$b	\$a es mayor o igual que \$b

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_2**" crearemos un archivo llamado **operadores\_comparacion.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de operadores de comparación en PHP 7</title>
</head>

<body>

<h2>Ejemplo de operadores de comparación</h2>
<?php

$a = 8;

$b = 3;

$c = 5;

if($a == $b)
{
    echo "$a es igual a $b <br>";
}
else
{
    echo "$a no es igual a $b <br>";
}

if($a != $b)
{
    echo "$a no es igual a $b <br>";
}
else
{
    echo "$a es igual a $b <br>";
}

if($a < $b)
{
    echo "$a es menor a $b <br>";
}
else
```

```
{
    echo "$a no menor a $b <br>";
}

if($a > $b)
{
    echo "$a es mayor a $b <br>";
}
else
{
    echo "$a no es mayor a $b <br>";
}

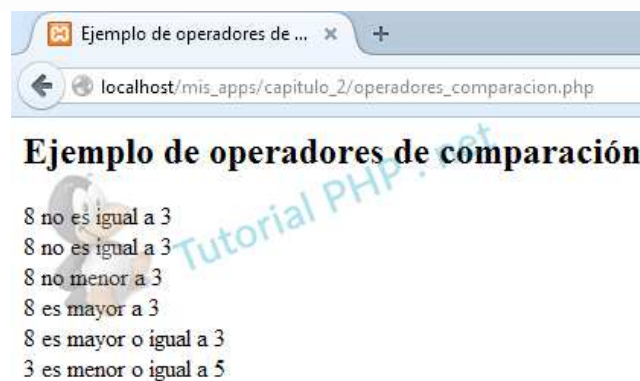
if($a >= $c)
{
    echo "$a es mayor o igual a $b <br>";
}
else
{
    echo "$a no es mayor o igual a $b <br>";
}

if($b <= $c)
{
    echo "$b es menor o igual a $c <br>";
}
else
{
    echo "$b no es menor o igual a $c <br>";
}

?>

</body>

</html>
```





## 2.3) Operadores lógicos en PHP 7

Los operadores lógicos en **PHP 7** son usados para evaluar varias comparaciones, combinando los posibles valores de estas. Veamos la siguiente tabla:

Operador	Nombre	Ejemplo	Devuelve cierto cuando
<b>&amp;&amp;</b>	Y	(7>2) && (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas
<b>and</b>	Y	(7>2) and (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas
<b>  </b>	O	(7>2)    (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera
<b>or</b>	O	(7>2) or (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera
<b>!</b>	No	!(7>2)	Niega el valor de la expresión

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_2**" crearemos un archivo llamado **operadores\_logicos.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de operadores de lógicos en PHP 7</title>
</head>

<body>

<h2>Ejemplo de operadores lógicos</h2>

<?php

$a = 8;

$b = 3;

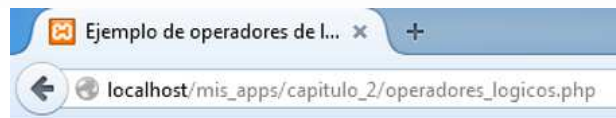
$c = 9;

if($a==8 && $b==3)
{
    echo "$a es igual a 8 y $b es igual a 3 <br>";
}
else
{
    echo "Alguna condición no cumplió la validación <br>";
}

if($a==8 and $b==3)
{
    echo "$a es igual a 8 y $b es igual a 3 <br>";
}
else
{
    echo "Alguna condición no cumplió la validación <br>";
}

if($a==8 || $c==5)
{
    echo "$a puede ser igual a 8 y $c puede ser igual a 5 <br>";
}
```

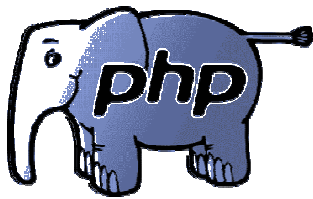
```
}  
else  
{  
    echo "Alguna condición no cumplió la validación <br>";  
}  
  
if($a==8 or $c==5)  
{  
    echo "$a puede ser igual a 8 y $c puede ser igual a 5 <br>";  
}  
else  
{  
    echo "Alguna condición no cumplió la validación <br>";  
}  
  
if($a!= 4)  
{  
    echo "$a no es igual a 4<br>";  
}  
else  
{  
    echo "Alguna condición no cumplió la validación <br>";  
}  
  
?>  
</body>  
</html>
```



## Ejemplo de operadores lógicos

8 es igual a 8 y 3 es igual a 3  
8 es igual a 8 y 3 es igual a 3  
8 puede ser igual a 8 y 9 puede ser igual a 5  
8 puede ser igual a 8 y 9 puede ser igual a 5  
8 no es igual a 4

## Capítulo 3.- Instrucciones en PHP 7



En el tercer de nuestro Tutorial PHP 7 veremos qué son las condicionales, los bucles, ¿qué es la función printf y el manejo de cadenas.

### 3.1) Condicionales en PHP 7

Las sentencias condicionales nos permiten ejecutar o no, determinadas instrucciones dependiendo del resultado a evaluar sobre una condición. Las más frecuentes son las siguientes instrucciones:

- if
- if / else
- if / elseif / else
- switch

Veamos algunos ejemplos:

#### if

La condicional **if** se emplea para evaluar una comparación siempre que sea verdadera, veamos este pseudocódigo:

```
<?php

if (condición)
{
    Sentencias a ejecutar cuando la condición es cierta.
}

?>
```

Veamos un ejemplo, creemos una carpeta dentro de "**mis\_apps**" que se llame "**capitulo\_3**" y en ella, creemos un archivo llamado **ejemplo\_condicional\_if.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de condicionales en PHP 7</title>
</head>

<body>

<h2>Ejemplo de condicionales</h2>
<?php

$a = 15;

// Equivale a Si $a es igual a 15

if ($a == 15)
{
    echo "$a vale 15";
}

?>

</body>

</html>
```



## if / else

La condicional **if / else** se emplea para evaluar una comparación siempre que sea verdadera y si no, se ejecutará lo que exista en el else. Veamos este pseudocódigo:

```
<?php

if (condición)
{
    Sentencias a ejecutar cuando la condición es cierta.
}
else
{
    Sentencias a ejecutar cuando la condición es falsa.
}

?>
```

Veamos un ejemplo, dentro de la carpeta "**mis\_apps/capitulo\_3**" creemos un archivo llamado **ejemplo\_condicional\_if\_else.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de condicionales en PHP 7</title>
</head>

<body>

<h2>Ejemplo de condicionales</h2>
<?php

$a = 15;

// Equivale a Si $a es igual a 13 sino imprimimos $a no vale 13

if ($a == 13)
{
    echo "$a vale 13";
}
else
{
    echo "$a no vale 13";
}

?>

</body>

</html>
```



### if / elseif / else

La condicional **if / elseif / else** se emplea para evaluar varias comparaciones siempre que sea verdadera y sino, se ejecutará lo que exista en el else. Veamos este pseudocódigo:

```
<?php

if (condición)
{
    Sentencias a ejecutar cuando la condición es cierta.
}
elseif (condición)
{
    Sentencias a ejecutar cuando la condición es cierta.
}
elseif (condición)
{
    Sentencias a ejecutar cuando la condición es cierta.
}
elseif (condición)
{
    Sentencias a ejecutar cuando la condición es cierta.
}
else
{
    Sentencias a ejecutar cuando la condición es falsa.
}

?>
```

Veamos un ejemplo, dentro de la carpeta "**mis\_apps/capitulo\_3**" creamos un archivo llamado **ejemplo\_condicional\_if\_elseif\_else.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de condicionales en PHP 7</title>
</head>

<body>

<h2>Ejemplo de condicionales</h2>
<?php

$a = 15;

// Equivale a Si $a es igual a 13, 14 o 15 sino imprimimos $a no vale eso
```

```
if ($a == 13)
{
    echo "$a vale 13";
}
elseif ($a == 14)
{
    echo "$a vale 14";
}
elseif ($a == 15)
{
    echo "$a vale 15";
}
else
{
    echo "$a no vale 13, 14 o 15";
}

?>

</body>

</html>
```



## switch

La condicional **switch** se emplea para evaluar varias comparaciones siempre que sea verdadera y sino, se ejecutará lo que exista en el else. Veamos este pseudocódigo:

```
<?php

switch (valor)
{
    case valor:
        Acción a realizar si se encontró el valor.
        break;

    case valor:
        Acción a realizar si se encontró el valor.
        break;

    case valor:
        Acción a realizar si se encontró el valor.
        break;

    case valor:
        Acción a realizar si se encontró el valor.
        break;

    case valor:
        Acción a realizar si se encontró el valor.
        break;
}
```

```
    default:
        Acción a realizar sino se encontró el valor.

}

?>
```

Veamos un ejemplo, dentro de la carpeta "**mis\_apps/capitulo\_3**" creemos un archivo llamado **ejemplo\_switch.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de condicionales en PHP 7</title>
</head>

<body>

<h2>Ejemplo de condicionales</h2>
<?php

$a = 5;

// Equivale a Si $a es igual a 13, 14 o 15 sino imprimimos $a no vale eso

switch ($a)
{
    case 0:
        echo "$a es igual a 0";
        break;

    case 1:
        echo "$a es igual a 1";
        break;

    case 2:
        echo "$a es igual a 2";
        break;

    case 3:
        echo "$a es igual a 3";
        break;

    case 4:
        echo "$a es igual a 4";
        break;

    case 5:
        echo "$a es igual a 5";
        break;

    default:
        echo "$a no es igual a 0, 1, 3, 4 ni 5";
}

?>

</body>

</html>
```





### 3.2) Qué son los bucles PHP 7

Los bucles nos permiten iterar conjuntos de instrucciones, es decir repetir la ejecución de un conjunto de instrucciones mientras se cumpla una condición. Existen cuatro tipos:

- While
- Do... While
- For
- Foreach

#### While

Su funcionamiento es sencillo, ya que primero se evalúa que la condición sea verdadera y luego se ejecuta, hasta que la condición pase a ser falsa; una sentencia while (Español: Mientras) puede que no se ejecute ni siquiera una vez, si su condición es inicialmente falsa. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_3**" crearemos un archivo llamado **bucle\_while.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de bucle en PHP 7</title>
</head>

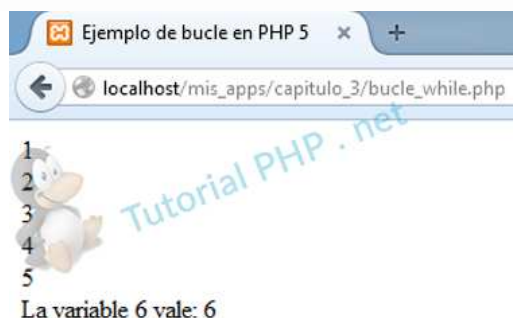
<body>

<?php
    $i = 1;

    while($i <= 5)
    {
        echo $i."<br>";
        $i += 1;
    }

    echo "La variable $i vale: ".$i."<br>";
?>

</body>
</html>
```



#### Do... While

Su uso es similar a while, pero aquí, las sentencias que siguen al do (Español: Hacer) se ejecutan por lo menos una vez y se comprueba la condición luego de la primera iteración; así, si es verdadera la condición se repite por segunda vez, si es falsa se continúa con las sentencias inmediatamente después de la instrucción while. Tiene sólo una sintaxis. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_3**" crearemos un archivo llamado **bucle\_do\_while.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de bucle en PHP 7</title>
</head>

<body>

<?php

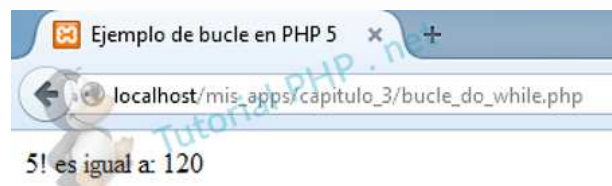
    $i = 5;
    $n = 1;

do
{
    $n = $n * $i;
    $i -= 1;
}

while($i > 1);

echo "5! es igual a: " . $n
?>

</body>
</html>
```



## For

Los bucles for (Español: Para) son los más complejos en PHP (y en otros lenguajes de programación). Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_3**" crearemos un archivo llamado **bucle\_for.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de bucle en PHP 7</title>
</head>

<body>

<?php

for($i = 1; $i <= 5; $i += 1) echo $i . "\n";

?>

</body>
</html>
```



## Foreach

Introducido en PHP 4 es una solución fácil para trabajar con arreglos, muy semejante a Perl y otros lenguajes, funciona solo en arreglos y presentara un error al utilizar una variable con diferente tipo o no inicializada. Existen dos sintaxis la segunda opción en menor pero tiene mejor uso que la primera. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_3**" crearemos un archivo llamado **bucle\_foreach.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de bucle en PHP 7</title>
</head>

<body>

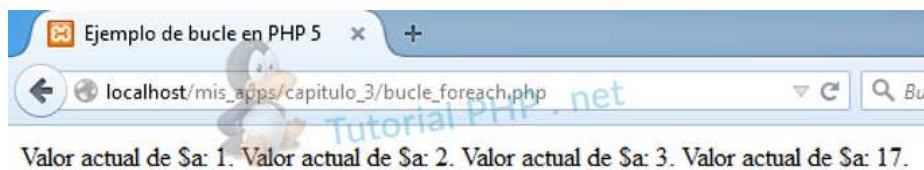
<?php

$a = array(1, 2, 3, 17);

foreach ($a as $v)
{
  echo "Valor actual de \$a: $v.\n";
}

?>

</body>
</html>
```



### 3.3) Salida función printf PHP 7

Hasta ahora hemos usado la instrucción **echo** para realizar salida a pantalla, esta instrucción es bastante limitada ya que no nos permite formatear la salida. En esta página veremos la instrucción **printf** que nos da mucha más potencia.

Veamos este pseudocódigo. Sentencia **printf()**:

```
<?php
printf(cadena formato, variable1, variable2, etc, etc);
?>
```

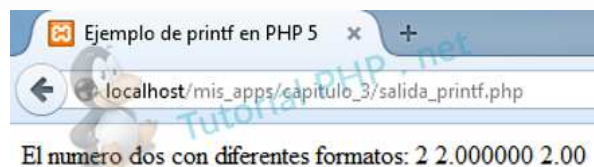
La cadena de formato indica cómo se han de representar las valores que posteriormente le indicaremos. La principal ventaja es que además de poder formatear los valores de salida, nos permite intercalar texto entre ellos. Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_3**" crearemos un archivo llamado **salida\_printf.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de printf en PHP 7</title>
</head>

<body>

<?php
printf("El numero dos con diferentes formatos: %d %f %.2f",2,2,2);
?>

</body>
</html>
```



La cadena de formato puede incluir una serie de caracteres especiales que indican como formatear las variables que se incluyen en la instrucción. Veamos la siguiente tabla:

Elemento	Tipo de variable
%s	Cadena de caracteres
%d	Número sin decimales
%f	Número con decimales
%c	Carácter ASCII

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps\capitulo\_3**" crearemos un archivo llamado **salida\_printf\_2.php** con el siguiente código:

```
<html>
<head>
  <title>Ejemplo de printf en PHP 7</title>
</head>

<body>
```

```
<?php

$var = "texto";

$num = 3;

printf("Puede fácilmente intercalar <b>%s</b> con números <b>%d</b> <br>", $var, $num);

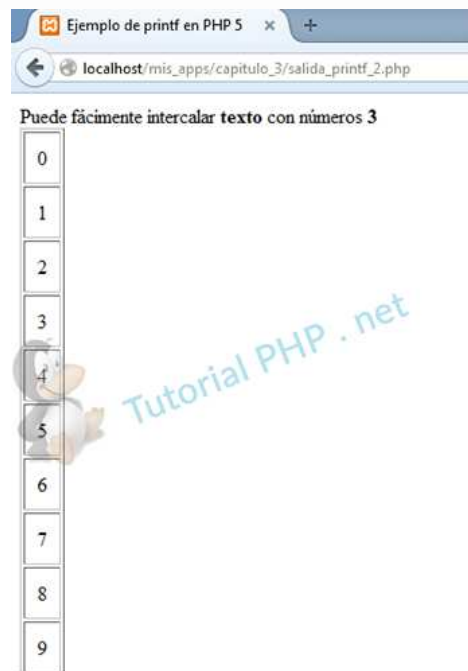
printf("<table border=1 cellpadding=10>");

for ($i=0; $i<10; $i++)
{
    printf("<tr><td>%10.d</td></tr>", $i);
}

printf("</table>");

?>

</body>
</html>
```



### 3.4) Manejo de cadenas PHP 7

Dado el uso del lenguaje PHP 7, el tratamiento de cadenas es muy importante, existen bastantes funciones para el manejo de cadenas, a continuación explicaremos las más usadas.

- **strlen(cadena)**. Nos devuelve el número de caracteres de una cadena.
- **explode(delimitador, string)**. Convierte en array la cadena mediante el delimitador.
- **sprintf(cadena de formato, var1, var2, etc, etc)**. Formatea una cadena de texto al igual que printf pero el resultado es devuelto como una cadena.
- **substr(cadena, inicio, longitud)**. Devuelve una subcadena de otra, empezando por inicio y de longitud.
- **chop(cadena) o rtrim(cadena)**. Elimina los saltos de línea y los espacios finales de una cadena.
- **strpos(cadena1, cadena2)**. Busca la cadena2 dentro de cadena1 indicándonos la posición en la que se encuentra.
- **str\_replace(cadena1, cadena2, texto)**. Reemplaza la cadena1 por la cadena2 en el texto.
- **ucfirst(cadena)**. Convierte el primer caracter de una cadena a mayúsculas.
- **ucwords(cadena)**. Convierte a mayúsculas el primer caracter de cada palabra de una cadena.
- **strtolower(cadena)**. Convierte una cadena a minúsculas.
- **strtoupper(cadena)**. Convierte un string a mayúsculas.
- **trim(cadena)**. Elimina espacio en blanco (u otro tipo de caracteres) del inicio y el final de la cadena.
- **md5(cadena)**. Calcula el hash md5 de un string.
- **nl2br(cadena)**. Inserta saltos de línea HTML antes de todas las nuevas líneas de un string.

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps/capitulo\_3**" crearemos un archivo llamado **cadenas.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de cadenas en PHP 7</title>
</head>

<body>

<?php

    // StrLen()
    echo "<strong>Ejemplo de Strlen()</strong> <br>".strlen("12345")."<br>";
    echo "<hr>";

    // Explode()
    $pieza = "una-dos-tres-cuatro-cinco";
    $piezas = explode("-", $pieza);

    echo "<strong>Ejemplo de Explode()</strong> <br>";

    foreach($piezas as $individuales)
    {
        echo $individuales."<br>";
    }
    echo "<hr>";

    // Sprintf()
    $num = 5;
    $objeto = 'árbol';
    $formato = 'Hay %d monos en el %s';

    echo "<strong>Ejemplo de Sprintf()</strong> <br>";
    echo sprintf($formato, $num, $objeto);
    echo "<hr>";
```

```
// Substr()
$cadenaSubstr = "Hola mundo. Esta es una cadena a evaluar.";
echo "<strong>Ejemplo de Substr()</strong> <br>";
$cadenaSubstr = substr($cadenaSubstr, 4, 10);
echo $cadenaSubstr."<br>";
echo "<hr>";

// Chop()
echo "<strong>Ejemplo de Chop()</strong> <br>";
$cadenaChop = "Hola mundo ";
echo "<pre>";
echo chop($cadenaChop);
echo "</pre >";
echo "<hr>";

// Strpos()
echo "<strong>Ejemplo de Strpos()</strong> <br>";
$cadenaStrpos = 'Hola mundo. Esta es una cadena a evaluar.';
$encontrar = 'mundo';
$pos = strpos($cadenaStrpos, $encontrar);

if ($pos === false)
{
    echo "Ops! la cadena <i>$encontrar</i> no fue encontrada en la cadena
<strong>$cadenaStrpos</strong>.";
}
else
{
    echo "La cadena <i>$encontrar</i> fue encontrada en la cadena
<strong>$cadenaStrpos</strong> y existe en la posición <strong>$pos</strong>.";
}
echo "<hr>";

// Str_replace()
echo "<strong>Ejemplo de Str_replace()</strong> <br>";

$vocales = array("a", "e", "i", "o", "A", "E", "I", "O");
$cadenSTRreplace = "Hola mundo. Esta es una cadena a evaluar.";
$reemplazador = array("4", "3", "1", "0", "4", "3", "1", "0");

$cadenSTRreplace = str_replace($vocales , $reemplazador, $cadenSTRreplace);

echo $cadenSTRreplace."<br>";
echo "<hr>";

// Ucfirst()
echo "<strong>Ejemplo de Ucfirst()</strong> <br>";

$cadenaUCfirst = 'hola mundo';
$cadenaUCfirst = ucfirst($cadenaUCfirst);

echo $cadenaUCfirst."<br>";
echo "<hr>";

// Ucwords()
echo "<strong>Ejemplo de Ucwords()</strong> <br>";

$cadenaUcwords = 'hola mundo';
$cadenaUcwords = ucwords($cadenaUcwords);

echo $cadenaUcwords."<br>";
echo "<hr>";
```



```
// Strtolower()
echo "<strong>Ejemplo de Strtolower()</strong> <br>";

$cadenaStrtolower = 'HOLA MUNDO';
$cadenaStrtolower = strtolower($cadenaStrtolower);

echo $cadenaStrtolower."<br>";
echo "<hr>";

// Strtoupper()
echo "<strong>Ejemplo de Strtoupper()</strong> <br>";

$cadenaStrtoupper = 'hola mundo';
$cadenaStrtoupper = strtoupper($cadenaStrtoupper);

echo $cadenaStrtoupper."<br>";
echo "<hr>";

// Trim()
echo "<strong>Ejemplo de Trim()</strong> <br>";
$cadenaTrim = "    Hola mundo    ";
echo "<pre>";
echo trim($cadenaTrim);
echo "</pre >";
echo "<hr>";

// Md5()
echo "<strong>Ejemplo de md5()</strong> <br>";
$cadenaMD5 = "Cadena cualquiera";
$cadenaMD5 = md5($cadenaMD5);
echo $cadenaMD5."<br>";
echo "<hr>";

// NL2br()
echo "<strong>Ejemplo de NL2br()</strong> <br>";
$cadenaNL2br = "Hola\nmundo\nHTML";
echo "<pre>";

$cadenaNL2br = nl2br($cadenaNL2br);
echo $cadenaNL2br;

echo "</pre >";
echo "<hr>";

?>

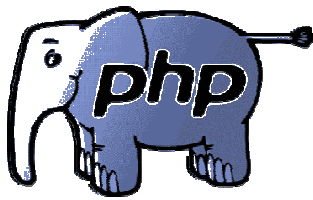
</body>
</html>
```



The screenshot shows a web browser window with the address bar displaying 'http://localhost/mis\_apps/capitulo\_3/cadenas.php'. The page content consists of several sections, each demonstrating a different PHP string function. A large, semi-transparent watermark 'Tutorial PHP . net' is overlaid diagonally across the page. A small cartoon penguin is visible in the background of the text.

```
Ejemplo de Strlen()  
5  
  
Ejemplo de Explode()  
una  
dos  
tres  
cuatro  
cinco  
  
Ejemplo de Sprintf()  
Hay 5 monos en el árbol  
  
Ejemplo de Substr()  
mundo Es  
  
Ejemplo de Chop()  
Hola mundo  
  
Ejemplo de Strpos()  
La cadena mundo fue encontrada en la cadena Hola mundo. Esta es una cadena a evaluar, y existe en la posición 5.  
  
Ejemplo de Str_replace()  
H0l4-mund0. 3st4 3s un4 c4d3n4 4 3v4h4r.  
  
Ejemplo de Ucfirsr()  
Hola mundo  
  
Ejemplo de Ucwords()  
Hola Mundo  
  
Ejemplo de Strtolower()  
hola mundo  
  
Ejemplo de Strtoupper()  
HOLA MUNDO  
  
Ejemplo de Trim()  
Hola mundo  
  
Ejemplo de md5()  
1a45d997bd3533d87082bfc58404533a  
  
Ejemplo de Nl2br()  
Hola  
mundo  
HTML
```

## Capítulo 4.- *Funciones en PHP 7*



En el cuarto capítulo de nuestro Tutorial PHP 7 veremos qué son las funciones, para qué nos sirve y como emplearlas.

## 4.1) ¿Qué son las funciones en PHP 7?

En el mundo de la programación hay dos maneras de escribir código. Una es escribiendo códigos largos, extensos y repitiendo pedazos del código. La otra es dividiendo el código en pequeñas partes que se puedan volver a usar sin que se tenga que repetir el mismo código una y otra vez. Obviamente la segunda manera es la correcta y aquí es donde entran en juego las funciones (function).

### ¿Qué es una Función?

Las funciones son básicamente pedazos de código que pueden ser llamados desde un script para realizar una tarea específica. A las funciones se les pueden pasar argumentos o parámetros de ser necesario para que utilicen sus valores para realizar alguna operación y retorna algún valor al final de la ejecución.

En PHP existen dos tipos de funciones, las que PHP trae por defecto para que el programador las utilice y las que el programador crea desde cero dependiendo de sus necesidades. Aquí abordaremos la segunda opción.

### ¿Cómo Escribir una Función en PHP?

El primer paso crear una función en PHP es ponerle un nombre con el cual se pueda hacer referencia a la misma. Las convenciones para los nombres de las funciones son las mismas que para las variables. Por lo tanto, el nombre de la función debe empezar con una letra o con un guión bajo (\_) y no se permiten espacios o signos de puntuación. Por último debes tener cuidado de que el nombre no sea el mismo que el de alguna función nativa de PHP.

Las funciones en PHP se crean usando la palabra clave `function` seguida por el nombre y por último un par de paréntesis (). El código que ejecutará la función es encerrado entre corchetes.

Veamos un ejemplo, creemos una carpeta dentro de "**mis\_apps**" que se llame "**capitulo\_4**" y en ella, creemos un archivo llamado **primer\_ejemplo\_funcion.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de funciones en PHP 7</title>
</head>

<body>

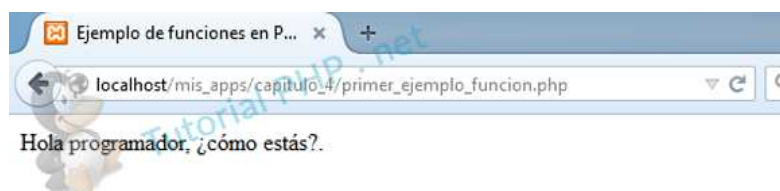
    <?php

        function saludame()
        {
            echo "Hola programador, ¿cómo estás?.";
        }

        // Invocamos la función
        saludame();

    ?>

</body>
</html>
```



## ¿Cómo Retornar un Valor de una Función?

Las funciones pueden retornar un valor al final de su ejecución si es que el programador lo necesita. Este valor puede ser de cualquier tipo y se hace con la sentencia `return`.

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps/capitulo\_4**", creamos un archivo llamado **segundo\_ejemplo\_funcion.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de funciones en PHP 7</title>
</head>

<body>

<?php

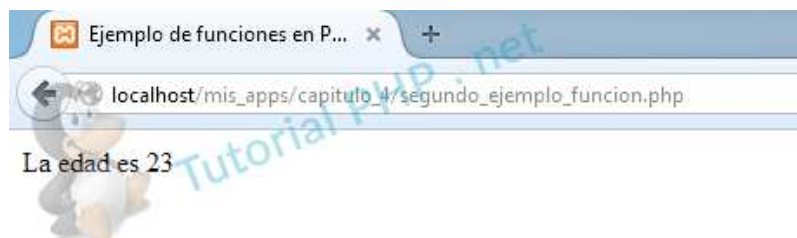
function validarEdad()
{
    $edad = 23;

    if($edad == 23)
    {
        return true;
    }
    else
    {
        return false;
    }
}

if(validarEdad())
{
    echo "La edad es 23";
}

?>

</body>
</html>
```



## Pasando Parámetros a la Función

Los parámetros o argumentos pueden ser pasados a una función y no hay limitación en cuanto al número de parámetros que puedan ser. Una función puede ser diseñada para aceptar parámetros mediante la colocación de los parámetros que se esperan dentro de los paréntesis que van después del nombre de la función. Los parámetros que escriben como si fueran variables y si son varios se deben separar con comas. En el siguiente ejemplo vamos a utilizar los parámetros `$dato1`, `$dato2`, `$dato3` e `$item`. Estos parámetros pueden ser utilizados dentro de la función como variables normales.

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps/capitulo\_4**", creamos un archivo llamado **tercer\_ejemplo\_funcion.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de funciones en PHP 7</title>
</head>

<body>

<?php

    function sumarDatos($dato1,$dato2,$dato3,$item)
    {

        $totalDeDatos = $dato1+$dato2+$dato3;

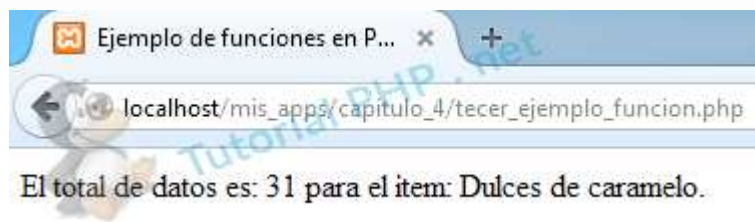
        echo "El total de datos es: ".$totalDeDatos." para el item: ".$item.".";

    }

    sumarDatos(10,20,1,'Dulces de caramelo');

?>

</body>
</html>
```



## Usando variables fuera de Función

Cuando tengamos que utilizar una variable que no esté declarada dentro de la función tendremos que hacer uso de global.

Veamos un ejemplo, dentro de nuestra carpeta "**mis\_apps/capitulo\_4**", creamos un archivo llamado **cuarto\_ejemplo\_funcion.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de funciones en PHP 7</title>
</head>

<body>

<?php

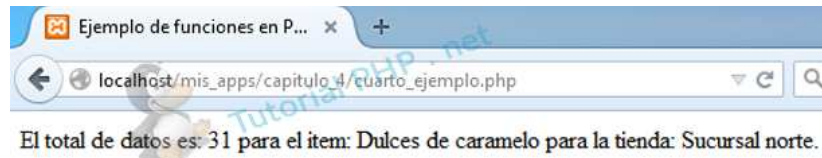
    // Variable fuera de la función
    $tienda = "Sucursal norte";

    function sumarDatos($dato1,$dato2,$dato3,$item)
    {

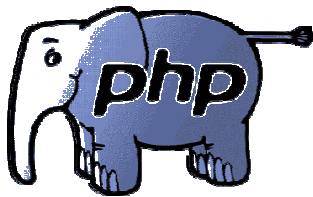
        // La hacemos variable global
        global $tienda;

        $totalDeDatos = $dato1+$dato2+$dato3;
```

```
    echo "El total de datos es: ".$totalDeDatos." para el item: ".$item." para la tienda: ".$tienda .".";  
  
    }  
  
    sumarDatos(10,20,1,'Dulces de caramelo');  
  
    ?>  
  
</body>  
</html>
```



## Capítulo 5.- *Procesado de Formularios en PHP 7*



En el quinto capítulo de nuestro Tutorial PHP 7 veremos cómo utilizar los formularios, y cómo recuperar los datos enviados en ellos, con las variables globales **\$\_POST** y **\$\_GET**.

También veremos cómo enviar correos electrónicos con la función **mail()** de **PHP 7**.



## 5.1) Recuperar datos en métodos GET y POST

**GET** y **POST** son dos métodos HTTP con los cuales podemos enviar datos de un formulario a nuestro PHP. Pero, ¿qué es **GET** y qué es **POST**? . Vamos a verlo.

### GET vs. POST

Estrictamente hablando, la diferencia entre **GET** y **POST** reside en cómo se transfieren las solicitudes. La información en **GET** se transmite en la URL. Si alguna vez has visto una URL que incluye signos de interrogación y los símbolos de unión, estabas buscando a una petición **GET**: **example.com/request.php?var1=foo&var2=bar** es una cadena **GET** que codifica la variable "var1" como "foo" y la variable "var2" como "bar".

La información **POST** está incluida en el cuerpo de la solicitud, y no es visible en la cadena de URL (y por lo tanto, no es visible para el usuario), ésta viaja "en un sobre" y llega al servidor.

Veamos un ejemplo, creemos una carpeta dentro de "**mis\_apps**" que se llame "**capitulo\_5**" y en ella, creemos un archivo llamado **ejemplo\_get.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de métodos HTTP en PHP 7</title>
</head>

<body>

    <?php

        if($_GET)
        {
            echo "El formulario se ejecutó con éxito.<br><br>";

            echo "<pre>";
            print_r($_GET);
            echo "</pre >";
        }

    ?>

    <form action="" method="get">

        <p>
            Dime cuál es tu nombre: <input type="text" name="nombre">
        </p>

        <p>
            Dime cómo se llama tu ciudad: <input type="text" name="ciudad">
        </p>

        <p>
            ¿En qué año naciste?:
            <select name="anoNacimiento">
                <option value="null">Selecciona un año</option>
                <?php
                    $anos = 1900;
                    while ($anos < 2000)
                    {
                        ?>
                        <option value="<?=$anos;?>"><?=$anos;?></option>
                        <?php
                            $anos++;
```

```

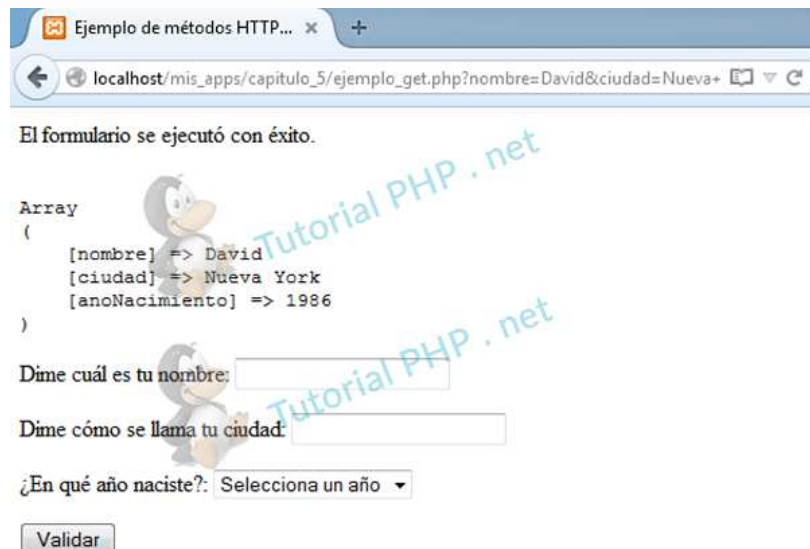
    }
    ?>
    </select>
</p>

<p>
    <input type="submit" value="Validar">
</p>

</form>

</body>
</html>

```



Ahora es el turno de **POST**. Veamos un ejemplo, creemos una carpeta dentro de "**mis\_apps**" que se llame "**capitulo\_5**" y en ella, creemos un archivo llamado **ejemplo\_post.php** con el siguiente código:

```

<html>
<head>
    <title>Ejemplo de métodos HTTP en PHP 7</title>
</head>

<body>

<?php

if($_POST)
{
    echo "El formulario se ejecutó con éxito.<br><br>";

    echo "<pre>";
    print_r($_POST);
    echo "</pre >";
}

?>

<form action="" method="post">

    <p>
        Dime cuál es tu nombre: <input type="text" name="nombre">
    </p>

    <p>

```

```

    Dime cómo se llama tu ciudad: <input type="text" name="ciudad">
  </p>

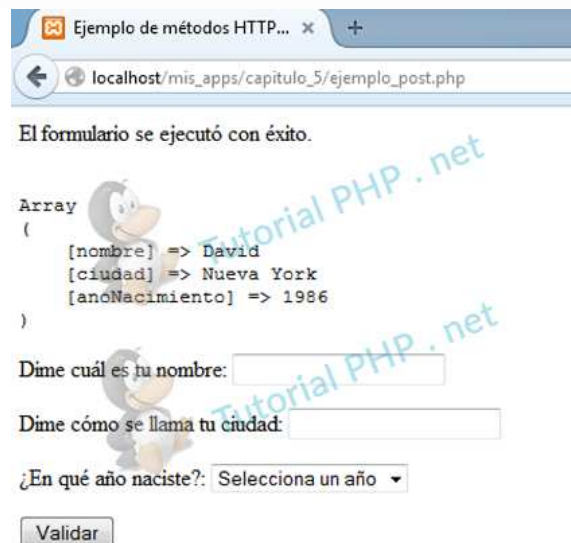
  <p>
    ¿En qué año naciste?:
    <select name="anoNacimiento">
      <option value="null">Selecciona un año</option>
      <?php
        $anos = 1900;
        while ($anos < 2000)
        {
          ?>
          <option value="<?=$anos;?>"><?=$anos;?></option>
          <?php
            $anos++;
          }
          ?>
        }
      </select>
    </p>

    <p>
      <input type="submit" value="Validar">
    </p>

  </form>

</body>
</html>

```



## 5.2) ¿Qué es la función mail en PHP 7?

PHP 7 nos ofrece la posibilidad de enviar correos electrónicos de una manera sencilla y muy fácil, para ello hacemos uso de la función **mail()**.

### Función email() en PHP 7

Veamos un ejemplo, dentro de "mis\_apps/capitulo\_5" creemos un archivo llamado **ejemplo\_mail.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de email en PHP 7</title>
</head>

<body>

<?php

if($_POST)
{
    // Correo al que queremos que Llegue
    $destinatario = "TU_CORREO_AQUI@gmail.com";
    // Asunto
    $asunto = "Email de prueba del Tutorial PHP 7";
    // Mensaje
    $mensaje = "Hola, este email es una prueba del Tutorial PHP 7. Los datos anexos al email
son: <br><br>
        Nombre: "._POST['nombre'].<br>
        Ciudad: "._POST['ciudad'].<br>
        Año de nacimiento: "._POST['anoNacimiento'].<br><br>

        Saludos!";
    // Cabeceras
    // Para enviar un correo HTML, debe establecerse la cabecera Content-type
    $cabeceras = 'MIME-Version: 1.0' . "\r\n";
    $cabeceras .= 'Content-type: text/html; charset=iso-8859-1' . "\r\n";
    // Cabeceras adicionales
    $cabeceras .= 'To: TU NOMBRE <TU_CORREO_AQUI@gmail.com>' . "\r\n";
    $cabeceras .= 'From: Tutorial PHP 7 <tutorial@tutorialphp.net>' . "\r\n";
    // Enviamos el email

    if(@mail($destinatario, $asunto, $mensaje, $cabeceras))
    {
        echo "El email se envió correctamente a ".$destinatario.".";
    }
    else{
        echo "El email no se pudo enviar.";
    }
}

?>

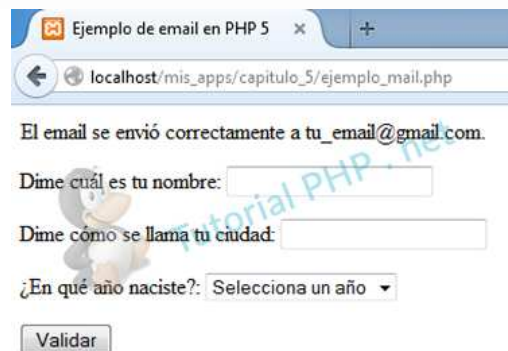
<form action="" method="post">
    <p>
        Dime cuál es tu nombre: <input type="text" name="nombre">
    </p>
```

```

<p>
  Dime cómo se llama tu ciudad: <input type="text" name="ciudad">
</p>
<p>
  ¿En qué año naciste?:
  <select name="anoNacimiento">
    <option value="null">Selecciona un año</option>
    <?php
      $anos = 1900;
      while ($anos < 2000)
      {
        ?>
        <option value="<?=$anos;?>"><?=$anos;?></option>
        <?php
          $anos++;
        }
      ?>
    </select>
  </p>
<p>
  <input type="submit" value="Validar">
</p>
</form>

</body>
</html>

```



## Email de prueba del Tutorial PHP 5

Recibidos x

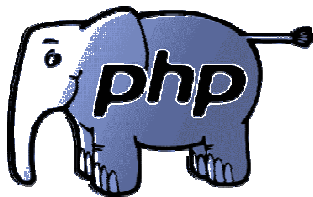
**Tutorial PHP 5** tutorial@tutorialphp.net  
para mí, mí

Hola, este email es una prueba del Tutorial PHP 5. Los datos anexos al email son:

Nombre: Gerardo  
Ciudad: Nueva York  
Año de nacimiento: 1989

Saludos!

## Capítulo 6.- *Bases de datos MySQL en PHP 7*



En el sexto capítulo de nuestro Tutorial PHP 7 veremos cómo interactuar con la **Base de Datos MySQL** desde **PHP 7**.

## 6.1) Crear la base de datos y tabla MySQL en PHP 7

Para la realización de este **Tutorial de PHP 7** con uso de Base de Datos hemos elegido al servidor **MySQL** por ser gratuita, y por ser también la más empleada en entornos **GNU/Linux y UNIX** (ya que comunmente **PHP 7** es más usado en estas plataformas).

Vamos a practicar, primero nos conectaremos al servidor **MySQL**:

```
mysql -u NUESTRO_USUARIO -p
```

Seguido, nos pedirá nuestra contraseña del usuario.

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 5.6.24 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Ya que estamos conectados, el comando para crear una base de datos es el siguiente:

```
create database NOMBREDELABASEDEDATOS;
```

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 23
Server version: 5.6.24 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database tutorialphp5;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Con este comando conseguimos crear la una base de datos en el servidor de bases de datos de nuestro servidor.

Una vez conseguido esto debemos crear las tablas en la base de datos, la descripción de las tablas contienen la estructura de la información que almacenaremos en ellas, para lo cual usaremos en lenguaje de consultas SQL común para todas las bases de datos relacionales.

En este ejemplo creamos una tabla llamada prueba con 3 campos: un campo identificador, que nos servirá para identificar unívocamente una fila con el valor de dicho campo, otro campo con el nombre de una persona y por último un campo con el apellido de la persona. Vamos a crearla.

Primero debemos indicarle al servidor MySQL con cual base de datos trabajemos:

```
USE NOMBREDELABASEDEDATOS;
```

Ahora sí, creamos la tabla:

```
CREATE TABLE personas (
  ID int(11) NOT NULL auto_increment,
  Nombre varchar(100),
  Apellidos varchar(100),
  PRIMARY KEY (ID),
  UNIQUE ID (ID)
);
```

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 5.6.24 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database tutorialphp5;
Query OK, 1 row affected (0.00 sec)

mysql> use tutorialphp5;
Database changed
mysql> CREATE TABLE personas (
->      ID int(11) NOT NULL auto_increment,
->      Nombre varchar(100),
->      Apellidos varchar(100),
->      PRIMARY KEY (ID),
->      UNIQUE ID (ID)
-> );
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Por último, para ver los campos de la tabla “personas”, ejecutamos el siguiente comando, nos tiene que devolver la descripción de la próxima imagen:

```
DESCRIBE personas;
```

```
C:\xampp\mysql\bin>mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 32
Server version: 5.6.24 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database tutorialphp5;
Query OK, 1 row affected (0.00 sec)

mysql> use tutorialphp5;
Database changed
mysql> CREATE TABLE personas (
->      ID int(11) NOT NULL auto_increment,
->      Nombre varchar(100),
->      Apellidos varchar(100),
->      PRIMARY KEY (ID),
->      UNIQUE ID (ID)
-> );
Query OK, 0 rows affected (0.00 sec)

mysql> describe personas;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| ID    | int(11) | NO   | PRI | NULL    | auto_increment |
| Nombre | varchar(100) | YES |     | NULL    |               |
| Apellidos | varchar(100) | YES |     | NULL    |               |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.02 sec)

mysql>
```

Muy bien, hemos aprendido a crear una base datos y una tabla en MySQL.



## 6.2) Conectarse a la Base de Datos de MySQL en PHP 7

Una vez que tenemos creada la base de datos en nuestro servidor, el siguiente paso es conectar-nos a la misma desde una página **PHP 7**. Para ello **PHP 7** nos proporciona una serie de instrucciones para acceder a bases de datos **MySQL**. Vamos a verlas:

Veamos un ejemplo, creemos una carpeta dentro de "**mis\_apps**" que se llame "**capitulo\_6** y en ella, creemos un archivo llamado **ejemplo\_conexion\_bd.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de conexión a base de datos MySQL</title>
</head>

<body>

<?php

// Dirección o IP del servidor MySQL
$host = "localhost";

// Puerto del servidor MySQL
$puerto = "3306";

// Nombre de usuario del servidor MySQL
$usuario = "root";

// Contraseña del usuario
$contrasena = "";

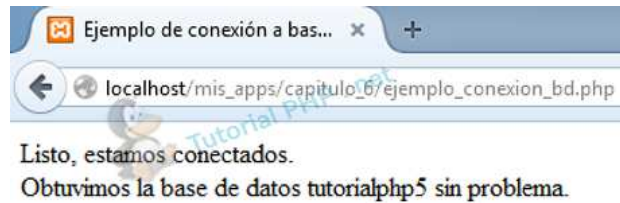
// Nombre de la base de datos
$baseDeDatos = "tutorialphp5";

// Nombre de la tabla a trabajar
$tabla = "personas";

function Conectarse()
{
    global $host, $puerto, $usuario, $contrasena, $baseDeDatos, $tabla;

    if (!($link = mysqli_connect($host.":".$puerto, $usuario, $contrasena)))
    {
        echo "Error conectando a la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Listo, estamos conectados.<br>";
    }
    if (!mysqli_select_db($link, $baseDeDatos))
    {
        echo "Error seleccionando la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Obtuvimos la base de datos $baseDeDatos sin problema.<br>";
    }
    return $link;
}
```

```
$link = Conectarse();  
  
mysql_close($link);  
  
?>  
  
</body>  
</html>
```



Al ejecutar la instrucción **mysql\_connect()** creamos un vínculo entre la base de datos y el código **PHP 7**, este vínculo será usado posteriormente en las consultas que hagamos a la base de datos.

Finalmente, una vez que hemos terminado de usar el vínculo con la base de datos, lo liberaremos con la instrucción **mysql\_close()** para que la conexión no quede ocupada.

### 6.3) Consultas a la Base de Datos MySQL en PHP 7

Una vez que nos hemos conectado con el servidor de bases de datos **MySQL** con **PHP 7**, ya podemos realizar consultas a las tablas almacenadas, así como sus filas.

Veamos un ejemplo, creemos un archivo llamado **ejemplo\_select\_mysql.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de selección de datos en base de datos MySQL</title>
</head>

<body>

    <?php

// Dirección o IP del servidor MySQL
$host = "localhost";

// Puerto del servidor MySQL
$puerto = "3306";

// Nombre de usuario del servidor MySQL
$usuario = "root";

// Contraseña del usuario
$contrasena = "";

// Nombre de la base de datos
$baseDeDatos = "tutorialphp5";

// Nombre de la tabla a trabajar
$tabla = "personas";

function Conectarse()
{
    global $host, $puerto, $usuario, $contrasena, $baseDeDatos, $tabla;

    if (!($link = mysqli_connect($host.":".$puerto, $usuario, $contrasena)))
    {
        echo "Error conectando a la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Listo, estamos conectados.<br>";
    }
    if (!mysqli_select_db($link, $baseDeDatos))
    {
        echo "Error seleccionando la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Obtuvimos la base de datos $baseDeDatos sin problema.<br>";
    }
    return $link;
}

$link = Conectarse();
```

```
$query = "SELECT Nombre, Apellidos FROM $tabla;";

$result = mysqli_query($link, $query);

?>

<table>
<tr>
    <td>Nombre</td>
    <td>Apellidos</td>
</tr>

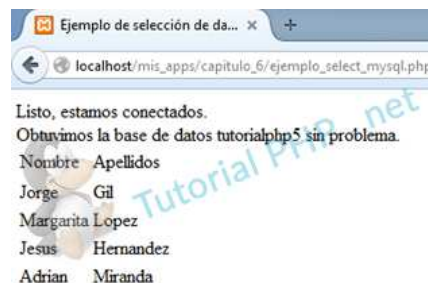
<?php
while($row = mysqli_fetch_array($result))
{
    printf("<tr><td>%s</td><td>%s</td></tr>", $row["Nombre"],$row["Apellidos"]);
}

mysqli_free_result($result);
mysqli_close($link);

?>

</table>

</body>
</html>
```



En este ejemplo hemos utilizado 3 instrucciones nuevas:

**mysqli\_query()**, **mysqli\_fetch\_array()** y **mysqli\_free\_result()**.

Con la instrucción **mysqli\_query()** hemos hecho una consulta a la base de datos en el lenguaje de consultas propio **SQL**, con la instrucción **mysqli\_fetch\_array()** extraemos los datos de la consulta a un array y con **mysqli\_free\_result()** liberamos la memoria usada en la consulta.

## 6.4) Insertar registros a la Base de Datos MySQL en PHP 7

Ya que vimos como conectar a la base de datos **MySQL** y obtener sus datos, veámos cómo insertar filas desde **PHP 7**.

Veamos un ejemplo, creemos un archivo llamado **ejemplo\_insert\_mysql.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de ingreso de datos en base de datos MySQL</title>
</head>

<body>

<?php

// Dirección o IP del servidor MySQL
$host = "localhost";

// Puerto del servidor MySQL
$puerto = "3306";

// Nombre de usuario del servidor MySQL
$usuario = "root";

// Contraseña del usuario
$contrasena = "";

// Nombre de la base de datos
$baseDeDatos ="tutorialphp5";

// Nombre de la tabla a trabajar
$tabla = "personas";

function Conectarse()
{
    global $host, $puerto, $usuario, $contrasena, $baseDeDatos, $tabla;

    if (!($link = mysqli_connect($host.":". $puerto, $usuario, $contrasena)))
    {
        echo "Error conectando a la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Listo, estamos conectados.<br>";
    }
    if (!mysqli_select_db($link, $baseDeDatos))
    {
        echo "Error seleccionando la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Obtuvimos la base de datos $baseDeDatos sin problema.<br>";
    }
    return $link;
}

$link = Conectarse();
```

```

if($_POST)
{
    $queryInsert = "INSERT INTO $tabla (Nombre, Apellidos) VALUES
('".$_POST['nombreForm']."', '".$_POST['apellidoForm']."')";

    $resultInsert = mysqli_query($link, $queryInsert);

    if($resultInsert)
    {
        echo "<strong>Se ingresaron los registros con exito</strong>. <br>";
    }
    else
    {
        echo "No se ingresaron los registros. <br>";
    }
}

$query = "SELECT Nombre, Apellidos FROM $tabla;";

$result = mysqli_query($link, $query);

?>

<table>
<tr>
<td>Nombre</td>
<td>Apellidos</td>
<tr>

<?php
while($row = mysqli_fetch_array($result))
{
    echo "<tr>";
    echo "<td>";
    echo $row["Nombre"];
    echo "</td>";
    echo "<td>";
    echo $row["Apellidos"];
    echo "</td>";
    echo "</tr>";
}

mysqli_free_result($result);

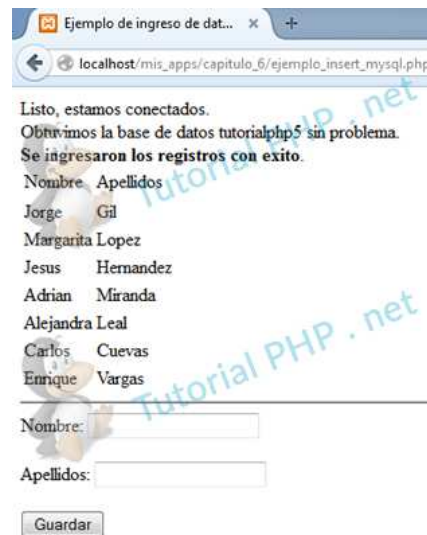
mysqli_close($link);

?>

</table>
<hr>
<form action="" method="post">
    Nombre: <input type="text" name="nombreForm"> <br> <br>
    Apellidos: <input type="text" name="apellidoForm"> <br> <br>
    <input type="submit" value="Guardar">
</form>

</body>
</html>

```



Ejemplo de ingreso de dat... x

localhost/mis\_apps/capitulo\_6/ejemplo\_insert\_mysql.php

Listo, estamos conectados.  
Obtuvimos la base de datos tutorialphp5 sin problema.  
Se ingresaron los registros con éxito.

Nombre	Apellidos
Jorge	Gil
Margarita	Lopez
Jesus	Hernandez
Adrian	Miranda
Alejandra	Leal
Carlos	Cuevas
Enrique	Vargas

---

Nombre:

Apellidos:

En nuestro script, si se ejecuta el método post (**\$\_POST**) se inicia el nuevo query e insertamos el registro con la instrucción **INSERT** del lenguaje de base de datos **SQL**.

## 6.5) Actualizar registros de Base de Datos MySQL en PHP 7

Hemos visto cómo conectar, seleccionar e insertar registros en la de base de datos **MySQL** con **PHP 7**. En este capítulo veremos como actualizarlo.

Veamos un ejemplo, creemos un archivo llamado **ejemplo\_update\_mysql.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de actualización de datos en base de datos MySQL</title>
</head>

<body>

<?php

// Dirección o IP del servidor MySQL
$host = "localhost";

// Puerto del servidor MySQL
$puerto = "3306";

// Nombre de usuario del servidor MySQL
$usuario = "root";

// Contraseña del usuario
$contrasena = "";

// Nombre de la base de datos
$baseDeDatos ="tutorialphp5";

// Nombre de la tabla a trabajar
$tabla = "personas";

function Conectarse()
{
    global $host, $puerto, $usuario, $contrasena, $baseDeDatos, $tabla;

    if (!($link = mysqli_connect($host.":".$puerto, $usuario, $contrasena)))
    {
        echo "Error conectando a la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Listo, estamos conectados.<br>";
    }
    if (!mysqli_select_db($link, $baseDeDatos))
    {
        echo "Error seleccionando la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Obtuvimos la base de datos $baseDeDatos sin problema.<br>";
    }
    return $link;
}

$link = Conectarse();
```



```

if($_POST)
{
    $queryUpdate = "UPDATE $tabla SET Nombre = '". $_POST['nombreForm']. "',
                    Apellidos = '". $_POST['apellidoForm']. "'
                    WHERE ID = '". $_POST['idForm']. "';";

    $resultUpdate = mysqli_query($link, $queryUpdate);

    if($resultUpdate)
    {
        echo "<strong>El registro ID ". $_POST['idForm']. " con exito</strong>. <br>";
    }
    else
    {
        echo "No se pudo actualizar el registro. <br>";
    }
}

$query = "SELECT ID, Nombre, Apellidos FROM $tabla;";

$result = mysqli_query($link, $query);

?>

<table>
    <tr>
        <td>Nombre</td>
        <td>Apellidos</td>
    <tr>

<?php
while($row = mysqli_fetch_array($result))
{
    echo "<tr>";
    echo "<td>";
    echo $row["Nombre"];
    echo "</td>";
    echo "<td>";
    echo $row["Apellidos"];
    echo "</td>";
    echo "<td>";
    echo "<a href='\"?id=\". $row[\"ID\"]. \">Actualizar</a>";
    echo "</td>";
    echo "</tr>";

}

mysqli_free_result($result);

?>

</table>
<hr>

<?php
if($_GET)
{
    $querySelectByID = "SELECT ID, Nombre, Apellidos FROM $tabla WHERE ID =
    '". $_GET['id']. "';";

```

```

$resultSelectByID = mysqli_query($link, $querySelectByID);

$rowSelectByID = mysqli_fetch_array($resultSelectByID);
?>

<form action="" method="post">
  <input type="hidden" value="<?=$rowSelectByID['ID'];?>" name="idForm">
  Nombre: <input type="text" name="nombreForm" value="<?=$rowSelectByID['Nombre'];?>">
<br> <br>
  Apellidos: <input type="text" name="apellidoForm" value="<?=$rowSelectByID['Apellidos'];?>"> <br> <br>
  <input type="submit" value="Guardar">
</form>

<?php
}
mysqli_close($link);
?>
</body>
</html>

```

Ejemplo de actualización d... x +

localhost/mis\_apps/capitulo\_6/ejemplo\_update.php?id=5

Listo, estamos conectados.  
Obtuvimos la base de datos tutorialphp5 sin problema.  
**El registro ID 5 con éxito.**

Nombre	Apellidos	
Jorge	Aceves	<a href="#">Actualizar</a>
Margarita	Lopez	<a href="#">Actualizar</a>
Jesus	Hernandez	<a href="#">Actualizar</a>
Adrian	Miranda	<a href="#">Actualizar</a>
Alejandra	Leal	<a href="#">Actualizar</a>
Carlos	Cuevas	<a href="#">Actualizar</a>
Enrique	Vargas	<a href="#">Actualizar</a>

---

Nombre:

Apellidos:

## 6.6) Borrar registros de la Base de Datos MySQL en PHP 7

Y finalmente, para cerrar el capítulo de **Bases de Datos MySQL en PHP 7** nos queda por ver el borrado de registros. El borrado de registros es uno de los procesos más sencillos.

Veamos un ejemplo, creemos un archivo llamado **ejemplo\_delete\_mysql.php** con el siguiente código:

```
<html>
<head>
    <title>Ejemplo de borrado de datos en base de datos MySQL</title>
</head>

<body>

<?php

// Dirección o IP del servidor MySQL
$host = "localhost";

// Puerto del servidor MySQL
$puerto = "3306";

// Nombre de usuario del servidor MySQL
$usuario = "root";

// Contraseña del usuario
$contrasena = "";

// Nombre de la base de datos
$baseDeDatos = "tutorialphp5";

// Nombre de la tabla a trabajar
$tabla = "personas";

function Conectarse()
{
    global $host, $puerto, $usuario, $contrasena, $baseDeDatos, $tabla;

    if (!($link = mysqli_connect($host.".".$puerto, $usuario, $contrasena)))
    {
        echo "Error conectando a la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Listo, estamos conectados.<br>";
    }
    if (!mysqli_select_db($link, $baseDeDatos))
    {
        echo "Error seleccionando la base de datos.<br>";
        exit();
    }
    else
    {
        echo "Obtuvimos la base de datos $baseDeDatos sin problema.<br>";
    }
    return $link;
}

$link = Conectarse();
```

```

if($_GET)
{
    $queryDelete = "DELETE FROM $tabla WHERE ID = ".$_GET['id'].>";
    $resultDelete = mysqli_query($link, $queryDelete);

    if($resultDelete)
    {
        echo "<strong>El registro se ha eliminado con exito</strong>.<br>";
    }
    else
    {
        echo "Hubo un problema borrando el registro.";
    }
}

$query = "SELECT ID, Nombre, Apellidos FROM $tabla;";
$result = mysqli_query($link, $query);

?>

<table>
    <tr>
        <td>Nombre</td>
        <td>Apellidos</td>
    <tr>

<?php
while($row = mysqli_fetch_array($result))
{
    echo "<tr>";
    echo "<td>";
    echo $row["Nombre"];
    echo "</td>";
    echo "<td>";
    echo $row["Apellidos"];
    echo "</td>";
    echo "<td>";
    echo "<a href='\"?id=".$_row["ID"]."\">Borrnar</a>";
    echo "</td>";
    echo "</tr>";
}

mysqli_free_result($result);

mysqli_close($link);

?>

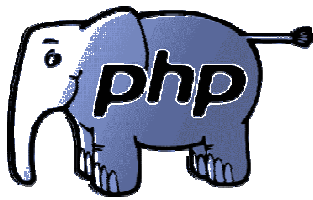
</table>
<hr>

</body>
</html>

```



## Capítulo 7.- Sesiones en PHP 7



En el séptimo capítulo de nuestro Tutorial PHP 7 veremos cómo usar las **sesiones** desde **PHP 7**.

## 7.1) ¿Qué son las sesiones en PHP 7?

En este artículo intentaré explicar, de la forma más simple posible, qué son las sesiones en **PHP 7**, cómo funcionan y cómo las implementa el servidor web. El objetivo de este artículo no es explicar cómo usar sesiones, sino comprender la mecánica del funcionamiento, para entender cómo actúan los ataques contra ellas.

### Antecedentes

El protocolo que utiliza la web (HTTP) es un protocolo sin estado, es decir, que no guarda ninguna información sobre conexiones anteriores: no sabe en qué página web estuviste antes ni si ya has enviado datos al sitio web. Lo único que conoce la página web son los datos que le llegan a través de la URL.

Dado que algunas páginas web querían poder reconocer a los visitantes y guardar datos de los mismos de una conexión a otra sin tener que ir arrastrando un montón de parámetros en la URL, se inventaron las ya famosas cookies. Una cookie es un fragmento de información que se guarda en el ordenador del usuario y que está asociado al navegador con el que visitó la página web. Esto quiere decir que cada navegador guarda sus propias cookies y no las comparte con otros navegadores. La forma en la que se guarda esta información depende del navegador, aunque muchos las almacenan como archivos de texto en un directorio dado.

Cuando un navegador va a abrir una página web, si tiene alguna cookie guardada asociada a ese sitio web (en realidad, a ese dominio), coge los datos de la misma y los manda al servidor web junto con la petición de la página.

El sitio web que pone una cookie, también le dice al navegador hasta cuando debe durar (como máximo) la misma: desde “hasta que se cierre el navegador” hasta cualquier cantidad de tiempo. Un dominio sólo puede acceder a las cookies que puso él mismo en el navegador, es decir, un dominio no puede acceder a las cookies de otros dominios.

### ¿Qué son las sesiones en PHP 7.

Resumiendo, las formas que tiene un sitio web para propagar datos sobre un visitante son: la URL o las cookies. El problema con ambos recursos es que son fácilmente modificables por el usuario: la URL se puede cambiar directamente en la barra de direcciones del navegador y las cookies, al ser datos en el ordenador del usuario, también pueden ser modificadas.

Así que, para guardar datos más críticos, se necesitaba un método que no fuera modificable por los usuarios (para evitar que, por ejemplo, un usuario autenticado se hiciera pasar por otro). Así nacieron las sesiones.

Una sesión en PHP 7 es una serie de caracteres aleatorios que forman una identificación única para cada visitante (a la que llamaremos “id de sesión”). Cuando a un usuario se le asigna un id de sesión, el servidor web crea un archivo en su sistema donde irá introduciendo todos los datos que queramos guardar. Pero, ¿cómo reconoce el sitio web al usuario en sucesivas conexiones? Pues usando alguno de los dos métodos anteriores; esto es, haciendo que el usuario le mande en la URL su id de sesión o enviándole una cookie con el mismo y con una duración de “hasta que se cierre el navegador”.

## ¿Cómo funcionan las sesiones en PHP 7?

Mejor lo explicamos con un ejemplo: acabamos de autenticarnos en una página con nuestro usuario (luis) y contraseña (\*\*\*\*\*). El sitio web comprueba que efectivamente son correctos y quiere poder reconocernos en las siguientes conexiones. Los datos que quiere guardar son: “usuario=luis”. No puede usar para esto la URL ya que entonces, cualquier persona que ponga en la barra de direcciones “usuario=luis” podría acceder a nuestra cuenta. Por la misma razón no puede usar cookies, ya que cualquier usuario sólo tendría que encontrar el archivo de su cookie y modificarla para que ponga “usuario=luis”. Así que lo que le queda son las sesiones.

Mediante PHP el servidor web le asigna a este usuario un id de sesión (por ejemplo: “31d7bgphebfbemb55311b1cger6”), crea en su sistema un archivo (en este ejemplo con el nombre “sess\_31d7bgphebfbemb55311b1cger6”) y mete dentro “usuario=luis”. Como este archivo no está en el ordenador del usuario, sino en el propio servidor, sólo puede ser modificado por el sitio web. Luego, el servidor le manda una cookie al navegador con nombre el nombre de la sesión (por defecto y para este ejemplo “PHPSESSID”) y valor el id de sesión (“31d7bgphebfbemb55311b1cger6”).

Así, cuando el usuario vaya a otra página dentro del mismo sitio web, le mandará al servidor el valor de su cookie: “PHPSESSID=31d7bgphebfbemb55311b1cger6” (este mismo efecto se puede conseguir utilizando la URL). De esta forma, el servidor sabe que el usuario es el que tiene id de sesión “31d7bgphebfbemb55311b1cger6”, así que va a buscar el archivo con sus datos (“sess\_31d7bgphebfbemb55311b1cger6”) y se los pasa a la página web.



## 7.2) Ejemplo de uso de sesiones en PHP 7

Vamos a ver un ejemplo para el uso de sesiones en **PHP 7**. En este caso usaremos los datos de entrada que el usuario nos indique y haremos uso de ellos.

Veamos un ejemplo, creemos una carpeta dentro de "**mis\_apps**" que se llame "**capitulo\_7**" y en ella, creemos un archivo llamado **ejemplo\_sesion.php** con el siguiente código:

```
<?php

if(session_id()=="")
{
    session_start();
}

if($_POST)
{
    if(@$_SESSION['autenticado']!=TRUE)
    {
        $_SESSION['autenticado'] = TRUE;
        $_SESSION['nombre'] = $_POST['nombreForm'];
        $_SESSION['edad'] = $_POST['edadForm'];
    }
}

if(@$_GET['salir']=="true")
{
    session_unset();
    session_destroy();
}

?>
<html>
<head>
    <title>Ejemplo de sesión en PHP 7</title>
</head>
<body>

<?php
if(@$_SESSION['autenticado']==TRUE)
{
    ?>
    <p>
        Hola <?=$_SESSION['nombre'];?>, tienes <?=$_SESSION['edad'];?> años y estás autentifi-
        cado. <a href="?salir=true">Haz click aquí para salir</a>.
    </p>
    <?php
    }
    else
    {
        ?>
        <p>
            <strong>¡Hola!, veo que no estás autenticado, por favor llena el formula-
            rio:</strong>
        </p>
        <form action="" method="post">
            Nombre: <input type="text" name="nombreForm"> <br> <br>
            Edad: <input type="text" name="edadForm"> <br> <br>
            <input type="submit" value="Guardar">
        </form>
    <?php
```

```

}
?>
</body>
</html>

```



Y con este último código, veremos cómo usar los datos. Creemos un archivo llamado **ejemplo\_sesion\_2.php** con el siguiente código:

```

<?php

if(session_id()=="")
{
    session_start();
}

?>

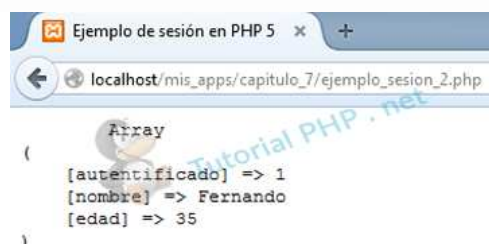
<html>
<head>
    <title>Ejemplo de sesión en PHP 7</title>
</head>
<body>

<pre>

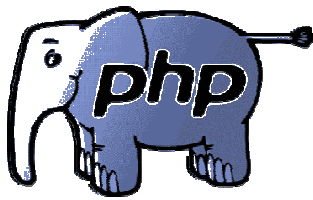
<?php
    print_r($_SESSION);
?>
</pre >

</body>
</html>

```



## Capítulo 8.- Cookies en PHP 7



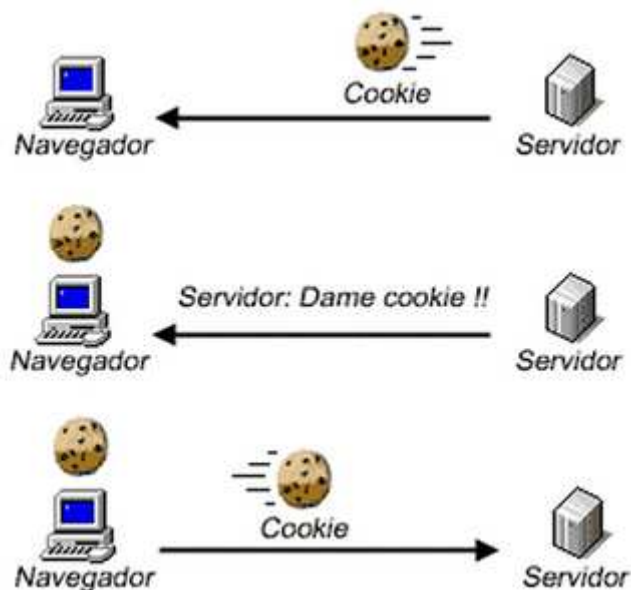
En el octavo capítulo de nuestro Tutorial PHP 7 veremos qué son las **Cookies** y cómo usarlas.

### 8.1) ¿Qué son las Cookies en PHP 7?

La principal utilidad de las **cookies** (galletas) en **PHP 7** es la de solventar el problema de la falta de estado en la navegación a través de las paginas web.

Con las cookies, pequeñas porciones de información se quedan registradas en el navegador permitiendo identificar a este a través de diferentes páginas de un mismo sitio e incluso durante visitas entre distintos días, muy ventajoso a comparación de las sesiones.

Realmente las cookies no son mas que cadenas de texto que son enviadas desde el servidor al cliente (navegador) y almacenadas en este, luego el navegador envía estas cookies al servidor permitiendo así la identificación del cliente en el servidor.



A continuación vamos a ver como usar las cookies para nuestro beneficio.

## 8.2) Ejemplos de cómo usar Cookies en PHP 7

El manejo de cookies en **PHP 7** se realiza mediante el uso de la función **setcookie()**, esta función esta disponible a partir de la versión **4 de PHP**.

```
<?php
    setcookie ( string $nombre [, string $valor [, int $expiracion = 0 [, string $path [,
string $dominio [, bool $secure = false [, bool $httponly = false ]]]]] )

    /*
    Todos Los argumentos excepto el nombre son opcionales.
    */
?>
```

**Setcookie()** define una cookie que es enviada junto con el resto de la información de la cabecera (header). Las **cookies PHP** deben ser enviadas antes de cualquier tag de html, js, css o espacios, por lo tanto deberemos realizar la llamada a estas funciones antes de cualquier tag. Esta es una restricción de las cookies no de **PHP 7**.

- **Nombre.** Nombre de la cookie. Si creamos una cookie solamente con el nombre, en el cliente se eliminara la cookie que exista con ese nombre. También podemos reemplazar cualquier argumento con una cadena vacía ("").
- **Valor.** Valor que almacenará la cookie en el cliente.
- **Expiración.** El argumento expire es un argumento entero que indica la hora en que se eliminara la cookie en el formato de hora que devuelven las funciones UNIX time() y mktime(). Normalmente se usa time() + N. segundos de duración, para especificar la duración de una cookie.
- **Path.** Subdirectorío en donde tiene valor la cookie.
- **Dominio.** Dominio en donde tiene valor la cookie. Si ponemos como dominio www.domain.com la cookie no se transmite para domain.com, mientras que si ponemos domain.com la cookie se transmite tanto para domain.com como para www.domain.com
- **Secure.** El argumento secure indica que la cookie solo se transmitirá a través de una conexión segura HTTPS.
- 

```
<?php

    setcookie("usuario", $_POST['nombreForm'], time()+3600, "/", $_POST['dominioForm']);

?>
```

En este ejemplo establecemos una cookie de nombre **usuario** que contiene el valor **Luis**, que dura **1 hora (3600 segundos)** y es válida para todo el dominio **tutorialphp.net**.

Veamos un ejemplo, creemos una carpeta dentro de "**mis\_apps**" que se llame "**capitulo\_8** y en ella, creemos un archivo llamado **ejemplo\_cookie.php** con el siguiente código:

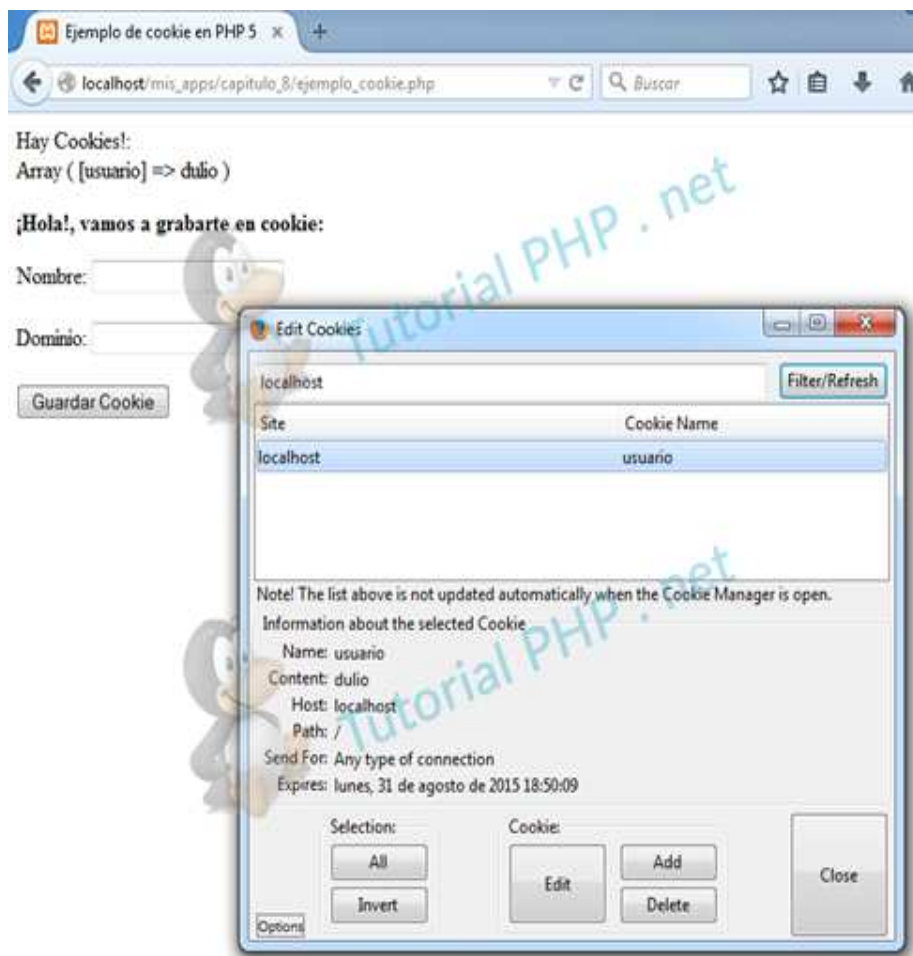
```
<?php
if($_POST)
{
    setcookie("usuario", $_POST['nombreForm'], time()+3600, "/", $_POST['dominioForm']);
}
?>

<html>
<head>
    <title>Ejemplo de cookie en PHP 7</title>
</head>
<body>
```

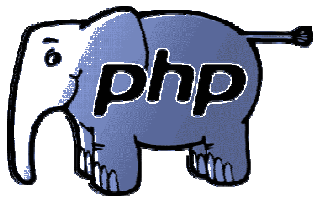
```

<?php
if($_COOKIE)
{
    echo "Hay Cookies!: <br>";
    print_r($_COOKIE);
}
else
{
    echo "No hay Cookies :( <br>";
}
?>
<p>
    <strong>¡Hola!, vamos a grabarte en cookie:</strong>
</p>
<form action="" method="post">
    Nombre: <input type="text" name="nombreForm"> <br> <br>
    Dominio: <input type="text" name="dominioForm"> <br> <br>
    <input type="submit" value="Guardar Cookie">
</form>
</body>
</html>

```



## Capítulo 9.- *Expresiones reguares EE.RR. en PHP 7*



Siendo éste el penúltimo capítulo de nuestro Tutorial PHP 7 veremos el tema de las **Expresiones Regulares**, qué son y cómo implementarlas en **PHP 7**.

### 9.1) ¿Qué son las expresiones regulares en PHP 7?

Las expresiones regulares son cadenas de caracteres que forman un patrón, expresan un texto más extenso y buscan una coincidencia, en **PHP 7** existen diversas funciones, vamos a ver las más importantes.

- **preg\_match()**.- Realiza una comparación con una expresión regular.
- **preg\_replace()**.- Realiza una búsqueda y sustitución de una expresión regular.
- **preg\_match\_all()**.- Realiza una comparación global de una expresión regular.
- **preg\_split()**.- Divide un string mediante una expresión regular.

Podemos encontrar más funciones para expresiones regulares en PHP 7 en el siguiente sitio oficial de PHP:

**<http://php.net/manual/es/ref.pcre.php>**



## 9.2) Veamos 10 ejemplos sobre patrones de expresiones regulares

En este artículo explicaré 10 ejemplos muy comunes sobre **expresiones regulares**, muy útiles para tus proyectos web 100% compatibles con funciones de **PHP 7**.

### 1) Validar una URL

¿Quieres saber si una dirección web es válida?, no hay problema, con esta expresión regular lo tendremos muy fácil:

```
/^(https?:\/\/)?([\da-z\.-]+\.[a-z]{2,6})([\/\w \?=-]*)*\:\/\/?$/
```

### 2) Validar un E-mail

En muchas ocasiones necesitaremos saber si un e-mail con el que se trata de registrar un usuario es válido:

```
^[a-z0-9-]+\.[a-z0-9-]+*@[a-z0-9-]+\.[a-z]{2,3}$
```

### 3) Comprobar la seguridad de una contraseña

Para aquellos que necesitáis sugerir / comprobar la fortaleza de una contraseña:

```
(?=^.{8,}$)((?=.*\d)|(?=.*\W+))(?![\.\n])(?=.*[A-Z])(?=.*[a-z]).*$
```

De esta forma comprobaremos:

- Contraseñas que contengan al menos una letra mayúscula.
- Contraseñas que contengan al menos una letra minúscula.
- Contraseñas que contengan al menos un número o caracter especial.
- Contraseñas cuya longitud sea como mínimo 8 caracteres.
- Contraseñas cuya longitud máxima no debe ser arbitrariamente limitada.

### 4) Validar un número de teléfono

Con este snippet se validarán todos los número de teléfono pertenecientes a los listados en la Wikipedia (en inglés):

```
^\+?\d{1,3}?[- .]?(\(?(\d{2,3})\)?[- .]?\d\d\d[- .]?\d\d\d\d$
```

### 5) Validar número de tarjeta de crédito

Ahora que tan de moda está el e-commerce seguro que esto le vendrá bien a más de uno:

```
^((67\d{2})|(4\d{3})|(5[1-5]\d{2})|(6011))(-?\s?\d{4}){3}((3[4,7])\s\d{2}-?\s?\d{6}-?\s?\d{5})$
```

## 6) Recoger ID de un vídeo de Youtube

Si necesitas recoger el ID de un vídeo de Youtube en sus múltiples combinaciones, esta es tu expresión regular:

```
/http:\/\/(?:youtu\.be\/|(?:[a-z]{2,3}\.)?youtube\.com\/watch(?:\?|#\!)v=)([w-]{11})\.*/gi
```

## 7) Validar todas las etiquetas de imagen sin cerrar

Ahora que prácticamente todos empleamos xHTML es interesante comprobar que todas nuestras etiquetas de imagen están correctamente cerradas:

```
<img([>]+)(\s*[^\s/])></img([>]
```

## 8) Validar una dirección IP

Si necesitas validar una dirección IP introducida por un usuario, esto te será de gran ayuda:

```
/^((([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5]))$/
```

## 9) Validar código postal

En muchas ocasiones necesitamos recoger en los formularios de alta el código postal:

```
^([1-9]{2}|[0-9][1-9]|[1-9][0-9])[0-9]{3}$
```

## 10) Validar un nombre de usuario

Si por ejemplo quisiéramos validar un nombre de usuario con un mínimo de 4 caracteres y un máximo de 15 haríamos lo siguiente:

```
/^[a-z\d_]{4,15}$/i
```

Además el nombre estaría utilizando sólo caracteres de la A-z y números.

### 9.3) Ejemplos de expresiones regulares

Ya que hemos visto una serie de ejemplos sumamente útiles de patrones de expresiones regulares, vamos a ver el siguiente ejemplo en donde ponemos en práctica algunas de ellas.

Veamos un ejemplo, creemos una carpeta dentro de "mis\_apps" que se llame "capitulo\_9" y en ella, creemos un archivo llamado **expresiones\_regulares.php** con el siguiente código:

<?php

```
/* Multi-Función */
function regexCheck($variable,$case)
{
    //Selector de la regex (expresión regular)
    switch ($case)
    {
        case "email":
            $regex = "/^[_a-z0-9-]+(\.[_a-z0-9-]+)*@[a-z0-9-]+(\.[a-z0-9-]+)*(\.[a-z]{2,3})$/";
            break;

        case "ipAddress":
            $regex = "/^(([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5]).){3}([1-9]?[0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$/";
            break;

        case "userName":
            $regex = "/^[a-z\d_]{4,15}$/i";
            break;
    }

    if(preg_match($regex,$variable))
    {
        return TRUE;
    }
    else
    {
        return FALSE;
    }
}

if(@$_POST['emailForm'])
{
    if(regexCheck($_POST['emailForm'], "email")){
        $resultadoEmail = "El email tiene el formato correcto.";
    }
    else
    {
        $resultadoEmail = "El email NO tiene el formato correcto.";
    }
}

if(@$_POST['IPForm'])
{
    if(regexCheck($_POST['IPForm'], "ipAddress")){
        $resultadoIP = "La dirección IP tiene el formato correcto.";
    }
    else
    {
        $resultadoIP = "La dirección IP NO tiene el formato correcto.";
    }
}
```

```
{
    $resultadoIP = "La dirección IP NO tiene el formato correcto.";
}

if(@$_POST['UserNameForm'])
{
    if(regexCheck($_POST['UserNameForm'], "userName")){
        $resultadoUN = "El nombre de usuario tiene el formato correcto.";
    }
    else
    {
        $resultadoUN = "El nombre de usuario tiene el formato correcto.";
    }
}

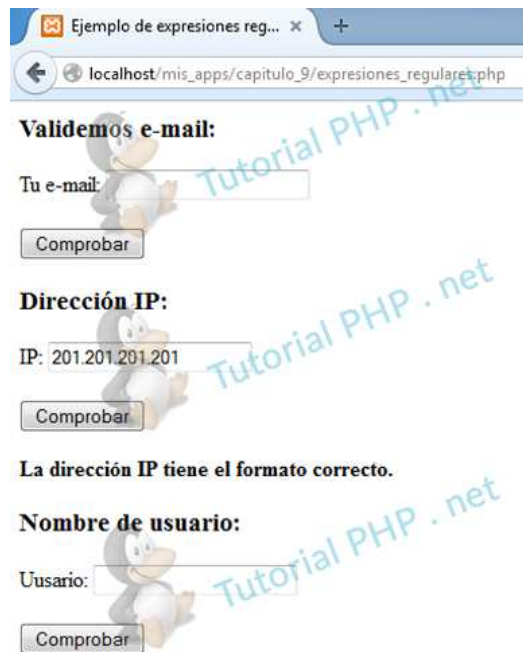
?>
<html>
    <head>
        <title>Ejemplo de expresiones regulares en PHP 7</title>
    </head>
    <body>

        <h3>Validemos e-mail:</h3>
        <form action="" method="post">
            Tu e-mail:
            <input type="text" name="emailForm"> <br> <br>
            <input type="submit" value="Comprobar">
        </form>
        <strong><?php echo @$resultadoEmail; ?></strong>

        <h3>Dirección IP:</h3>
        <form action="" method="post">
            IP:
            <input type="text" name="IPForm"> <br> <br>
            <input type="submit" value="Comprobar">
        </form>
        <strong><?php echo @$resultadoIP; ?></strong>

        <h3>Nombre de usuario:</h3>
        <form action="" method="post">
            Uusario:
            <input type="text" name="UserNameForm"> <br> <br>
            <input type="submit" value="Comprobar">
        </form>
        <strong><?php echo @$resultadoUN; ?></strong>

    </body>
</html>
```



Ejemplo de expresiones reg... x +

localhost/mis\_apps/capitulo\_9/expresiones\_regulares.php

**Validemos e-mail:**

Tu e-mail:

Comprobar

**Dirección IP:**

IP: 201.201.201.201

Comprobar

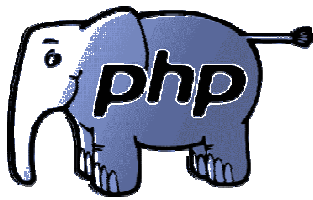
La dirección IP tiene el formato correcto.

**Nombre de usuario:**

Usuario:

Comprobar

## Capítulo 7.- Sesiones en PHP 7



En el séptimo capítulo de nuestro Tutorial PHP 7 veremos cómo usar las **sesiones** desde **PHP 7**.

## 10.1) ¿Qué son las clases en PHP 7?

Las siglas de **Programacion Orientada a Objetos (OOP en ingles – Object Oriented Programming – )** es una variante de programación que permite reutilizar gran parte del código, lo hace más estético y legible ya que está más orientado a la lógica humana, además de ser muy útil en grandes proyectos.

En proyectos con varios programadores, es posible asignar clases específicas que hagan un determinado trabajo a cada programador, y este desarrollarla independientemente de los avances de los demás, ya que los objetos son independientes unos de otros, esto hace más rápido y práctico el desarrollo del proyecto.

### OOP en PHP

PHP desde su versión 3.0 soportaba la sintaxis de la OOP sin embargo en ese tiempo (y un poco ahora) carecía de todas sus características. A este tiempo PHP soporta todo lo que un lenguaje orientado a objetos en teoría debería soportar:

- Encapsulamiento.
- Tipos Abstractos de Datos y ocultamiento de la Información.
- Herencia.
- Polimorfismo.

**PHP 7** no es un lenguaje 100% Orientado a Objetos sin embargo los soporta y muy bien, ¿por qué no obtener provecho de eso?, para comprender la POO, debemos comprender las partes que la conforman:

### ¿Qué son las clases?

La clase es una plantilla que usamos para crear objetos, al crear un objeto de una clase se dice que es una instancia de esa clase. Los objetos en PHP 7 se crean (instancian) con la palabra reservada **New**.

### ¿Qué son las clases?

La clase es una plantilla que usamos para crear objetos, al crear un objeto de una clase se dice que es una instancia de esa clase. Los objetos en PHP 7 se crean (instancian) con la palabra reservada **New**.

```
<?php
class Ejemplo
{
    //Constructor
    function Ejemplo()
    {
        //Nada por aquí
    }
}

//Iniciamos la clase
$obj = new Ejemplo();
?>
```

Aquí la variable **\$obj** sería una instancia de la clase **Ejemplo**.

## Constructor

El constructor es el método que es llamado automáticamente al crear una instancia de la clase, el constructor lleva el mismo nombre de la clase, el constructor no debe retornar ningún valor y es único (distintamente a C plus). Pero bien, para que una clase sea útil necesita métodos y atributos

## Métodos

Los métodos (también operaciones) no son más que un tipo de funciones propias de la clase, se manejan igual, reciben parámetros, desarrollan un proceso y devuelven (o no) un valor, los métodos se definen con la palabra reservada **function**.

```
<?php

class Ejemplo
{
    //Constructor
    function Ejemplo()
    {
        //Nada por aquí
    }

    function MuestraNombre($nombre)
    {
        echo "El nombre es " . $nombre;
    }
}

//Iniciamos la clase
$obj = new Ejemplo();

//Llamamos al método
$obj->MuestraNombre('Fernando Gil');

?>
```

Esto es una clase con un simple método que muestra el valor de su único parámetro programado, el ejemplo daría como resultado:

**El nombre es Fernando Gil**



## 10.2) Los atributos de POO en PHP 7

En este artículo veremos algo súper importante dentro de **PHP 7** orientado a objetos, los **Atributos**. Pero, ¿qué son los **Atributos**?, vamos a verlo.

Basicamente se les llama **Atributos** a variables que contienen información del estado de un objeto. Estos se definen usando la palabra reservada **var**.

Veamos el siguiente código:

```
<?php

class Ejemplo()
{
    var $atributo;

    function Ejemplo()
    {
        //Do nothing...
    }

    function DaValor($valor)
    {
        $this->atributo = $valor;
    }
}

$obj = new Ejemplo();

$obj->DaValor(4);

?>
```

Ahora el atributo valdra 4, \$this usada en el ejemplo, es una variable especial referenciada al objeto y atributos de la clase o de la clases heredadas, esto nos lleva al siguiente punto.

### 10.3) ¿Qué es la Herencia POO en PHP 7?

La **Herencia** permite a una clase, “heredar”, los **métodos** y **atributos** (ambos ya vistos en capítulos anteriores) de otra clase, para este proceso es usada la palabra reservada **extends** seguida de la que será su clase “**padre**”.

Vamos a ver el siguiente ejemplo:

```
<?php
class Ejemplo {

    var $atributo;

    function Ejemplo()
    {
        // Do nothing
    }

    function setAtributo($value)
    {
        $atributo = $value;
    }
}

class Ejemplo2 extends Ejemplo
{
    function Ejemplo2()
    {

    }

    function getAtributo()
    {
        return $atributo;
    }
}

$obj = new Ejemplo2();

$obj->setAtributo(7);

echo $obj->getAtributo();

?>
```

Como ves se pueden usar los métodos de las dos clases ya que al instanciar un objeto de la clase **Ejemplo2** al ser heredada de **Ejemplo** adquiere sus métodos.

## 10.4) Veamos el Acceso Public (Público) en POO PHP 7

### Acceso Public (Publico)

Al ser declarado un método o atributo **public**, éste será accesible desde cualquier punto del script, éste es el punto más bajo del sistema de proteccion. Para declarar público, se debe de usar la palabra reservada **public**.

```
<?php

class Ejemplo ()
{
    public $atributo;

    function Ejemplo()
    {
        $this->atributo = 'Hey!';
    }
}

$obj = new Ejemplo();

echo $obj->atributo;

?>
```

## 10.5) Veamos el Acceso Private (Privado) en POO PHP 7

### Acceso Private (Privado)

Este es el más restrictivo, los elementos declarados bajo **private** sólo serán accesibles desde métodos de su clase original.

```
<?php

class Ejemplo ()
{
    private $atributo;

    function Ejemplo()
    {
        $this->atributo = 'Hey!';
    }

    function daAtributo()
    {
        return $this->atributo;
    }
}

$obj = new Ejemplo();

echo $obj->daAtributo;
```

?>  
Como te darás cuenta, hemos tenido que instanciar un método para devolver el valor de atributo.

### Acceso Protected (Protegido)

Con este los elementos serán accesibles desde su clase original o desde clases heredadas.