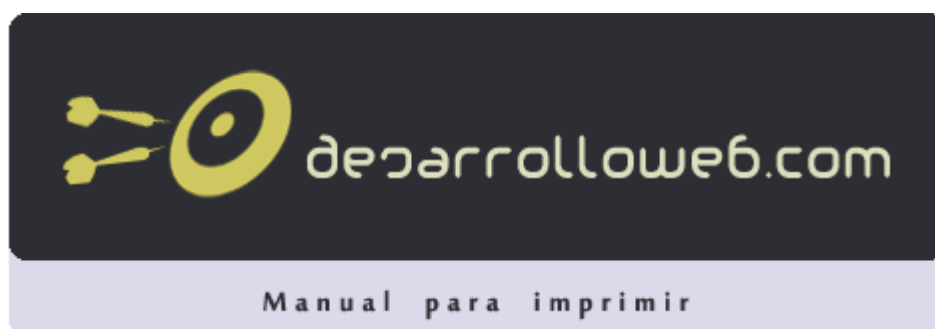


# Programación en PHP

*Principios básicos para la programación en PHP, el popular lenguaje del lado del servidor.  
Manual asequible para no programadores que sienta los fundamentos básicos de este lenguaje.  
Continuación lógica del manual de páginas dinámicas.*



## Autores del manual

Este manual ha sido realizado por los siguientes colaboradores de DesarrolloWeb.com:

**Rubén Alvarez**  
Redactor de DesarrolloWeb.com  
<http://www.desarrolloweb.com>  
(24 capítulos)

**Miguel Angel Alvarez**  
Director de DesarrolloWeb.com  
<http://www.desarrolloweb.com>  
(21 capítulos)

**Carlos Luis Cuenca**  
<http://www.helloworldsolutions.com/>  
(1 capítulo)

**Daniel López**  
<http://pichongol.blogspot.com>  
(1 capítulo)

**Jorge Ramos**  
(1 capítulo)

**Manu Gutierrez**  
<http://www.tufuncion.com>  
(1 capítulo)

# Parte 1:

# Qué es PHP

Capítulos introductorios donde hablaremos sobre los lenguajes de desarrollo del lado del servidor en general para explicar PHP en particular y que se entienda cuál es su modo de funcionamiento y los tipos de cosas que se pueden hacer con este lenguaje.

## 1.1.- Introducción a la programación en PHP

*Explicamos someramente qué es el PHP y lo comparamos a otros lenguajes para el desarrollo de webs dinámicas*

PHP es el lenguaje de lado servidor más extendido en la web. Nacido en 1994, se trata de un lenguaje de creación relativamente reciente, aunque con la rapidez con la que evoluciona Internet parezca que ha existido toda la vida. Es un lenguaje que ha tenido una gran aceptación en la comunidad de desarrolladores, debido a la potencia y simplicidad que lo caracterizan, así como al soporte generalizado en la mayoría de los servidores de hosting.

PHP nos permite embeber su pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz, combinando lo que ya sabemos del desarrollo HTML. Es decir, con PHP escribimos scripts dentro del código HTML, con el que se supone que ya estamos familiarizados. Por otra parte, y es aquí donde reside su mayor interés con respecto a los lenguajes pensados para los CGI, PHP ofrece un sinfín de funciones para la explotación de bases de datos de una manera llana, sin complicaciones.

Podríamos efectuar la quizás odiosa comparación de decir que PHP y ASP son lenguajes parecidos en cuanto a potencia y dificultad si bien su sintaxis puede diferir sensiblemente. Algunas diferencias principales pueden, no obstante, mencionarse:

**Actualizado:** En estos momentos ya no es tan polémica la comparación de PHP con ASP, puesto que son dos lenguajes que han evolucionado de maneras distintas. Mientras que ASP se ha estancado y han salido productos nuevos como .NET para sustituirlo, PHP ha ido mejorando mucho con los años y actualmente su potencia y posibilidades son totalmente distintas, con lo que ha dejado muy atrás la competencia con ASP. Este manual lo comenzamos con la versión 3 de PHP y hoy ya van por la 5 y están cerca de sacar la versión 6. Así pues ya no tiene mucho sentido comparar PHP con ASP, aunque las líneas siguientes a esta nota, que distinguen ASP de PHP, pueden ser de utilidad y una referencia válida, puesto que estas diferencias no han cambiado a día de hoy.

Así mismo, queremos informar que a pesar del manual tener ya cierto tiempo publicado, siempre lo estamos actualizando cuando surgen cambios en los modos de trabajo con PHP.

- PHP, aunque multiplataforma, ha sido concebido inicialmente para entornos UNIX y es en este sistema operativo donde se pueden aprovechar mejor sus prestaciones. ASP, siendo una tecnología Microsoft, está orientado hacia sistemas Windows, especialmente NT.
- Las tareas fundamentales que puede realizar directamente el lenguaje son definidas en PHP como funciones mientras que ASP invoca más frecuentemente los objetos. Por supuesto, esto no es más que una simple cuestión de forma ya que ambos lenguajes soportan igualmente ambos procedimientos.
- ASP realiza numerosas tareas sirviéndose de componentes (objetos) que deben ser comprados a determinadas empresas especializadas (o programados por nosotros mismos en otros lenguajes). PHP presenta una filosofía

totalmente diferente y, con un espíritu más generoso, es progresivamente construido por colaboradores desinteresados que implementan nuevas funciones en nuevas versiones del lenguaje.

Este manual va destinado a aquellos que quieren comenzar de cero el aprendizaje de este lenguaje y que buscan en él la aplicación directa a su proyecto de sitio o a la mejora de su sitio HTML. Los capítulos son extremadamente simples, sino simplistas, buscando ser accesibles a la mayoría. Ellos pueden ser complementados posteriormente con otros [artículos de mayor nivel](#) destinados a gente más experimentada.

La forma en la que hemos redactado este manual lo hace accesible a cualquier persona no familiarizada con la programación. Sin embargo, es posible que en determinados momentos alguien que no haya programado nunca pueda verse un poco desorientado. Nuestro consejo es el de no querer entender todo antes de pasar al siguiente capítulo sino intentar asimilar algunos conceptos y volver atrás en cuanto una duda surja o hayamos olvidado algún detalle. Nunca viene mal leer varias veces lo mismo hasta que quede bien grabado y asimilado.

Antes de comenzar a leer este manual es altamente aconsejable, sino imprescindible, haber leído previamente el manual sobre [manual sobre páginas dinámicas](#) en el cual se explica a grandes rasgos qué es el PHP, algunos conceptos útiles sobre el modo de trabajar con páginas dinámicas al mismo tiempo que nos introduce algunos elementos básicos de la programación como pueden ser las variables y las funciones.

Otra referencia a la cual haremos alusión es el [tutorial de SQL](#) que nos será de gran ayuda para el tratamiento de bases de datos y a MySQL, del que podremos aprender muchas cosas en el [Taller de MySQL](#).

Para todos los lectores, pero aun más para las personas más inexpertas y con más dificultades de aprendizaje, tenemos además una recomendación que puede ayudarles mucho. Se trata del [Videotutorial de PHP](#) que estamos publicando con diversos vídeos que explican con gran detalle la programación en PHP.

Esperamos que este manual resulte de vuestro agrado y que corresponda a nuestras expectativas: El poder acercar PHP a todos aquellos amantes del desarrollo de webs que quieren dar el paso hacia las webs "profesionales".

Los scripts que usamos en estos primeros ejemplos pueden ser descargados [aquí](#).

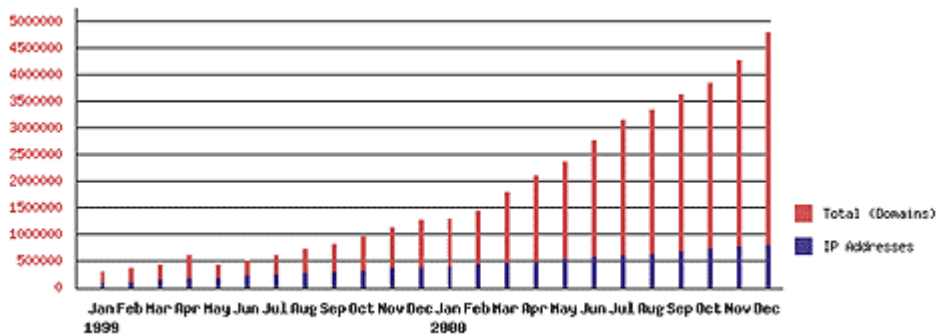
*Artículo por Rubén Álvarez*

## 1.2.- Breve historia de PHP

*Desde sus inicios hasta la versión 4 ha sido desarrollado por muchas personas.*

PHP es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.

La siguiente gran contribución al lenguaje se realizó a mediados del 97 cuando se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP versión 3. Actualmente PHP se encuentra en su versión 4, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de la anterior versión. Algunas mejoras de esta nueva versión son su rapidez -gracias a que primero se compila y luego se ejecuta, mientras que antes se ejecutaba mientras se interpretaba el código-, su mayor independencia del servidor web -creando versiones de PHP nativas para más plataformas- y un API más elaborado y con más funciones.



Gráfica del número de dominios y direcciones IP que utilizan PHP.  
Estadística de Netcraft.

En el último año, el número de servidores que utilizan PHP se ha disparado, logrando situarse cerca de los 5 millones de sitios y 800.000 direcciones IP, lo que le ha convertido a PHP en una tecnología popular. Esto es debido, entre otras razones, a que PHP es el complemento ideal para que el tándem Linux-Apache sea compatible con la programación del lado del servidor de sitios web. Gracias a la aceptación que ha logrado, y los grandes esfuerzos realizados por una creciente comunidad de colaboradores para implementarlo de la manera más óptima, podemos asegurar que el lenguaje se convertirá en un estándar que compartirá los éxitos augurados al conjunto de sistemas desarrollados en código abierto.

Artículo por Miguel Angel Alvarez

## 1.3.- Tareas principales del PHP

*Mencionamos los principales grupos de funciones integradas en el lenguaje y lo que ellas nos ofrecen.*

Poco a poco el PHP se va convirtiendo en un lenguaje que nos permite hacer de todo. En un principio diseñado para realizar poco más que un contador y un libro de visitas, PHP ha experimentado en poco tiempo una verdadera revolución y, a partir de sus funciones, en estos momentos se pueden realizar una multitud de tareas útiles para el desarrollo del web:

### 1.3.1.- Funciones de correo electrónico

Podemos con una facilidad asombrosa enviar un e-mail a una persona o lista parametrizando toda una serie de aspectos tales como el e-mail de procedencia, asunto, persona a responder...

Otras funciones menos frecuentes pero de indudable utilidad para gestionar correos electrónicos son incluidas en su librería.

### 1.3.2.- Gestión de bases de datos

Resulta difícil concebir un sitio actual, potente y rico en contenido que no es gestionado por una base de datos. El lenguaje PHP ofrece interfaces para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft, a partir de las cuales podremos editar el contenido de nuestro sitio con absoluta sencillez.

### 1.3.3.- Gestión de archivos

Crear, borrar, mover, modificar...cualquier tipo de operación más o menos razonable que se nos pueda ocurrir puede ser realizada a partir de una amplia librería de funciones para la gestión de archivos por PHP. También podemos transferir archivos por FTP a partir de sentencias en nuestro código, protocolo para el cual PHP ha previsto también gran cantidad de funciones.

### 1.3.4.- Tratamiento de imágenes

Evidentemente resulta mucho más sencillo utilizar Photoshop para una el tratamiento de imágenes pero...¿Y si tenemos que tratar miles de imágenes enviadas por nuestros internautas?

La verdad es que puede resultar muy tedioso uniformar en tamaño y formato miles de imágenes recibidas día tras día. Todo esto puede ser también automatizado eficazmente mediante PHP.

También puede parecer útil el crear botones dinámicos, es decir, botones en los que utilizamos el mismo diseño y solo cambiamos el texto. Podremos por ejemplo crear un botón haciendo una única llamada a una función en la que introducimos el estilo del botón y el texto a introducir obteniendo automáticamente el botón deseado.

A partir de la librería de funciones graficas podemos hacer esto y mucho más.

Muchas otras funciones pensadas **para Internet** (tratamiento de cookies, accesos restringidos, comercio electrónico...) o para **propósito general** (funciones matemáticas, explotación de cadenas, de fechas, corrección ortográfica, compresión de archivos...) son realizadas por este lenguaje. A esta inmensa librería cabe ahora añadir todas las funciones personales que uno va creando por necesidades propias y que luego son reutilizadas en otros sitios y todas aquellas intercambiadas u obtenidas en foros o sitios especializados.

Como puede verse, las posibilidades que se nos presentan son sorprendentemente vastas. Lo único que se necesita es un poco de ganas de aprender y algo de paciencia en nuestros primeros pasos. El resultado puede ser muy satisfactorio.

*Artículo por Rubén Álvarez*

## Parte 2: Cómo instalar PHP y MySQL

Explicaremos diversos modos que existen para instalar PHP y la base de datos MySQL, que es todo lo que necesitamos para empezar a trabajar. Para ello veremos cómo instalar PHP sobre los servidores web Apache o IIS. Además presentaremos diversos paquetes que nos permiten tener una instalación de todas las aplicaciones necesarias sin necesidad de ninguna configuración, lo que puede facilitarnos mucho la vida.

### 2.1.- Instalación de PHP en nuestro servidor

*Pasos previos a la programación. Instalación del modulo PHP en distintos servidores web.*

Como todo lenguaje de lado servidor, PHP, requiere de la instalación de un servidor en nuestro PC para poder trabajar en local. Este modo de trabajo resulta a todas luces más práctico que colgar los archivos por FTP en el servidor y ejecutarlos desde Internet.

Así pues, antes comenzar a crear nuestros programas en PHP, es necesario:

- Convertir nuestro ordenador en un servidor. Esto se hace instalando uno de los varios servidores disponibles para el sistema operativo de nuestra máquina.
- Introducir en nuestro servidor los archivos que le permitirán la comprensión del PHP. Estos archivos pueden ser descargados, en su versión más actual, de la [página oficial de PHP](#).

Para conocer la forma de instalar PHP sobre cada servidor de cada sistema operativo podemos dirigirnos al apartado de [documentación de la página oficial de PHP](#) donde disponemos de un manual en HTML de rápida consulta y un enorme manual en PDF de casi 1000 páginas traducido al castellano donde explican minuciosamente y entre otras cosas, los pasos a seguir para cada caso particular. De todos modos, nosotros vamos a ofrecer algunas ayudas para configurar PHP en los sistemas más habituales.

La elección de vuestro programa servidor tendrá mucho que ver con el sistema operativo que tengáis corriendo en vuestro ordenador. Estas serían algunas posibilidades de sistemas operativos y soluciones que funcionan bien.

### 2.1.1.- Windows 95/98

Si estáis trabajando en Windows 95 o Windows 98 y para desarrolladores principiantes, podría ser recomendable utilizar el servidor [Personal Web Ser.](#) En este caso necesitaríais:

- Personal Web Server de Microsoft como servidor el cual os sirve además para el aprendizaje en ASP. Tenéis una [guía de instalación y configuración](#) en esta misma web.
- Una instalación autoextraíble de la versión más reciente de PHP que, además de tardar menos en descargarse, os guiará paso a paso en el proceso de instalación. Esta versión no incluye todas las funcionalidades de PHP, pero os servirá para aprender hasta un buen nivel.

Hay que señalar que, para el caso de PHP en PWS, además de todo lo dicho en capítulo de instalación, es importante al crear el directorio virtual permitir la ejecución de scripts validando la caja correspondiente.

En Windows 95/98 también podremos utilizar el servidor Apache y puede que sea una opción todavía más completa que la de utilizar PWS. A continuación explicamos más sobre ello.

### 2.1.2.- Windows ME y XP Home edition

No hemos probado PHP en estas plataformas, pero en principio no tienen compatibilidad con Personal Web Server, por lo que deberíamos decantarnos por otro servidor.

Otra posibilidad para los usuarios de Windows en general es instalar [Apache](#) como servidor web lo cual puede resultar ventajoso con respecto al uso del PWS ya que PHP está principalmente diseñado para correr en este servidor. Esto quiere decir que, aunque en principio todo debería funcionar correctamente sobre ambos servidores, es posible que algún bug no corregido haga fallar uno de nuestros scripts si trabajamos para con un servidor cuyas actualizaciones son menos frecuentes y detalladas.

Apache ha sido especialmente pensado para plataformas Unix-Linux, aunque recientemente, con la Apache 2.0, han desarrollado una versión específica para Windows.

Disponemos de un artículo para aprender a [configurar PHP sobre Apache en Windows, como CGI](#) y también como [módulo de Apache](#).

### 2.1.3.- Windows NT, Windows 2000 y XP en sus versiones Profesional y Server

Para estos sistemas tenemos dos posibilidades muy interesantes, ya que podremos instalar PHP sobre [Internet Information Server](#) o sobre [Apache](#) con todas las garantías. Si hubiese que recomendar una de las dos opciones, nos decantaríamos por Apache debido a que, como decíamos, PHP está pensado para trabajar sobre Apache. Podría ser interesante IIS en el caso de que deseemos correr ASP y PHP sobre el mismo servidor, ya que, en principio, Apache no es compatible con ASP.

## 2.1.4.- Unix - Linux

Hay que decir, no obstante, que las mejores prestaciones de este lenguaje son obtenidas trabajando en entorno Unix o Linux y con un servidor Apache, la combinación más corriente en la mayoría de los servidores de Internet que trabajan con PHP.

## 2.1.5.- Conclusión

En cualquier caso, para fines de desarrollo en local, podemos contentarnos en un principio de trabajar con cualquier sistema. Solamente en casos de programación realmente avanzada podremos confrontarnos con problemas relacionados con el sistema operativo utilizado o el servidor en el que hacemos correr nuestras páginas. Hay que pensar también que, en casos puntuales para los que nuestro PC pueda quedarse corto, podemos hacer directamente nuestras pruebas en el servidor donde alojamos nuestro sitio el cual será muy probablemente, como hemos dicho, un Unix o Linux funcionando con Apache.

**Referencia:** En DesarrolloWeb hemos publicado diversos manuales y artículos que pueden ser una buena referencia para la instalación de PHP. Algunos se pueden ver a continuación:

[Configuración de PHP con Apache en Windows](#)  
[Configuración de PHP como módulo de Apache, también en Windows](#)  
[Instalación del Personal Web Server](#)  
[Instalación de IIS en Windows XP profesional](#)  
[Directorio de Apache](#) (Hemos publicado un manual o estará en breve)  
[FAQ sobre cómo instalar PHP en Windows](#)  
[Videotutorial de instalación de PHP con Wamp](#)  
[Video: instalar PHP en una llave USB](#)

*Artículo por Rubén Álvarez*

## 2.2.- Configuración de PHP con Apache en Windows, como CGI

*Aprende cómo configurar PHP y Apache para que trabajen conjuntamente en un sistema Windows. Instalación como CGI.*

El presente artículo trata de cómo **configurar PHP y Apache** para que trabajen conjuntamente en un sistema **Windows**. Además, este artículo asume que hay un servidor Apache configurado en el Windows, y que funciona correctamente.

**Nota:** Si deseamos conocer las distintas posibilidades para la instalación de PHP en los distintos sistemas operativos y servidores, puede ser de utilidad la lectura del artículo [Instalación de PHP en nuestro servidor](#).

Existen dos formas de configurar PHP para trabajar con Apache, instalar como un módulo o instalar como un CGI. En este artículo vamos a ver cómo instalarlo como CGI, aunque disponemos de otro artículo para [instalar PHP como módulo en Apache](#).

### 2.2.1.- Para instalar PHP como un CGI hay que seguir los siguientes pasos:

En primer lugar, hay que descargarse PHP desde la página de php.net. Existen dos versiones, una que tiene un instalador, y otra que es un fichero ZIP. Hay que descargarse esta última.

Una vez descargado, hay que descomprimirlo dentro de una carpeta, esta no tiene que estar bajo el árbol de directorios de Apache. El artículo asumirá que se descomprime dentro de la carpeta C:PHP. Comprobar que los contenidos del archivo ZIP no quedan en un subdirectorio de la carpeta C:PHP, sino directamente en dicha carpeta.

Dentro de la carpeta c:PHP se encuentra un fichero llamado PHP4ts.dll, hay que mover el fichero dentro de la carpeta:



c:\windowssystem ó c:\winntsystem

En este fichero se encuentra toda la configuración de PHP, y las modificaciones en la configuración de PHP (mostrar Errores, variables globales etc...) se encuentra dentro del mismo.

Es muy recomendable cambiar la directiva `display_errors` que por defecto esta en OFF, y ponerla en ON, para poder ver los errores que se producen en las páginas durante el desarrollo. Para un servidor en producción es conveniente dejarla en OFF.

Una vez se han hecho estos cambios, queda indicarle al Apache, donde se encuentra instalado el PHP, para ello hay que editar el fichero `httpd.conf` que se encuentra dentro de la carpeta `conf`, en la carpeta de instalación del apache (por defecto `c:\archivos de programa\apache group\apache2\conf`)

Abrir el fichero, y situarse al final del mismo, y escribir las siguientes líneas:

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

En ellas se indica donde se encuentra el ejecutable de php, y lo asocia a los ficheros `.php` que se encuentren dentro de apache.

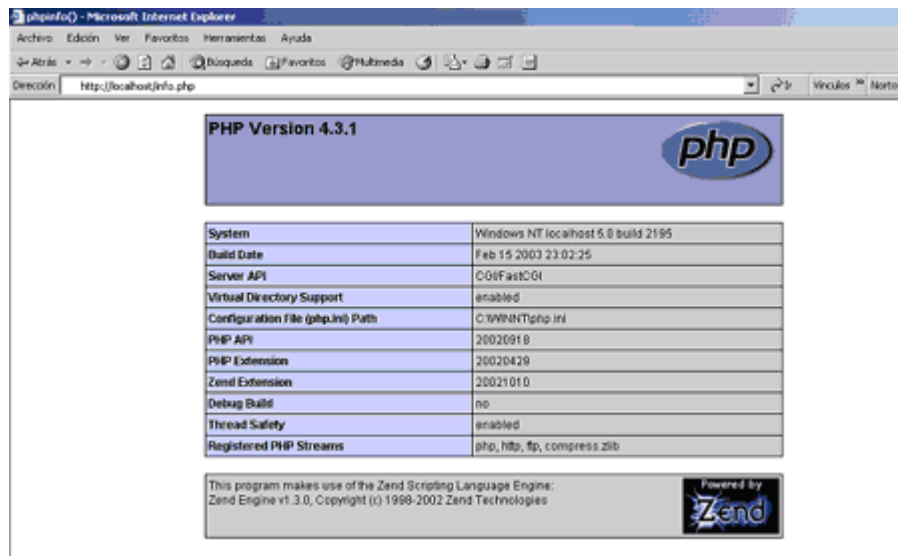
A continuación reiniciar el servidor Apache, y ya esta!

Por último, indicar que para probar la nueva instalación, es recomendable crear un fichero `php` con el siguiente contenido:

```
<? phpinfo();?>
```

Luego lo guardamos dentro de la carpeta raíz de documentos del Apache (por defecto `c:\archivos de programa\apache group\apache2\htdocs`), con un nombre terminado en `.php`, por ejemplo `info.php`

Para ejecutarlo, a través de un navegador, escribir la dirección `http://localhost/info.php`. Debería aparecer una pantalla como la que se muestra a continuación.



Si la vemos correctamente es que todo ha ido bien y que tenemos perfectamente instalado PHP en nuestro servidor Apache.

**Nota:** Este tipo de instalación de PHP sobre Apache es poco recomendada por motivos de seguridad. Podemos seguirla para configurar PHP en local, con intención de desarrollar nuestras páginas web, pero nunca si vamos a utilizar el servidor en un entorno de producción, es decir, en un servidor que se encuentre permanentemente conectado a Internet sirviendo páginas a todo tipo de usuarios.

Sería necesario [instalar PHP como un módulo de Apache](#), en lugar de CGI, para dotar al sistema de una mayor seguridad, y también más potencia.

**Referencia:** En esta FAQ damos otras opciones para la instalación de PHP, utilizando programas que permiten instalar y configurar Apache + PHP + MySQL en un sencillo paso, accesible para todos los usuarios: [Cómo instalar PHP en Windows](#).



Artículo por Carlos Luis Cuenca

## 2.3.- Configuración de PHP como modulo de Apache en Windows

*Explicamos el proceso completo para instalar PHP en una máquina Windows y un servidor Apache. La instalación se realiza como módulo que es lo más seguro y rápido.*

En este artículo vamos a explicar cómo instalar PHP como módulo de Apache 2.0 en un sistema Windows. Para las pruebas hemos utilizado Windows XP, pero seguro que con otros sistemas el proceso será muy parecido, aunque, en todo caso, indicaremos las diferencias documentadas en el sitio de PHP.

Anteriormente habíamos explicado la [instalación de PHP como un CGI](#), aunque en la página de PHP desaconsejan esta opción, puesto que adolece de graves problemas de seguridad. Además, PHP instalado como módulo de Apache resulta mucho más rápido que como CGI.

**Referencia:** Vamos a suponer que el servidor de páginas web Apache 2.0 está instalado en nuestro sistema. No obstante, para los que no lo tengan, les referimos a nuestro [manual de instalación y configuración de Apache](#).

### 2.3.1.- Descargar y descomprimir PHP

El primer paso consiste en descargar la última versión de PHP. Podremos hacerlo desde la página oficial de PHP, en la sección de descargas. <http://www.php.net/downloads.php> Debemos elegir la versión "zip package" que contiene todas las funcionalidades de PHP y el módulo necesario para instalarlo en Apache.

Una vez descargado el paquete comprimido en .zip de PHP necesitamos descomprimirlo en nuestro disco duro. Podemos utilizar el directorio raíz del disco duro para descomprimir los archivos. En ese caso, se creará un directorio llamado algo como "php-4.3.1-Win32" que colgará de nuestro directorio raíz. Se recomienda cambiar el nombre del directorio creado a algo como "c:php". En todo caso, nos advierten en la página de PHP sobre no colocar ningún nombre de directorio que contenga espacios, pues algún servidor web puede dar problemas. Por ejemplo, cuidado con instalar PHP en un directorio como este "c:archivos de programaphp", pues en la ruta tenemos directorios con espacios.

### 2.3.2.- Copia de las DLL

A continuación nos informan sobre la necesidad de copiar en nuestro directorio de sistema una serie de librerías (.dll), que encontraremos en el directorio sapi de nuestra instalación de PHP, supuestamente algo como "c:phpsapi",

El mencionado directorio de sistema puede variar de unas versiones a otras de Windows. En Windows XP, el directorio de sistema donde debemos copiar las dll, es "C:WINDOWSsystem32". En Windows 9x/ME, el directorio sería "C:WindowsSystem" y en Windows NT/2000 sería el directorio "C:WINNTSystem32" o bien, "C:WINNT40System32".

**Nota:** no se deben mezclar las DLL de diversas versiones de PHP, porque de lo contrario, podría causarnos problemas.

### 2.3.3.- Definir un archivo php.ini

Otro archivo que debemos copiar, esta vez en nuestro directorio Windows, es el php.ini, que guarda las opciones de configuración definidas para PHP. En la distribución de PHP se incluyen dos archivos php.ini que podemos utilizar directamente en nuestro sistema. Estos dos archivos se llaman "php.ini-dist" y "php.ini-recommended" y contienen unas opciones típicas de configuración de PHP. Se recomienda utilizar "php.ini-recommended", porque viene optimizado para obtener los mejores niveles de seguridad. En cualquier caso, podemos editar en cualquier momento el contenido del archivo

para modificar la configuración de PHP a nuestro gusto o necesidades.

Para definir el php.ini debemos hacer una copia del archivo de configuración escogido ("php.ini-dist" o "php.ini-recommended") y renombrarlo como el "php.ini". Posteriormente debemos copiarlo en nuestra carpeta Windows, que en sistemas 9x/ME/XP es "c:windows" y en sistemas NT/2000 suele ser "c:WINNT", o bien "c:WINNT40".

### 2.3.4.- Editar httpd.conf

Posteriormente deberemos editar nuestro archivo de configuración de Apache, llamado "httpd.conf" que está en el directorio "conf" de nuestra instalación de Apache. También podemos encontrar un acceso directo para editar este archivo accediendo a Inicio - Programas - Apache HTTP Server - Configure Apache HTTP Server - Edit httpd.conf configuration file.

Debemos añadir un par de líneas de configuración del módulo de Apache.

```
LoadModule php4_module C:\phpsapi\php4apache2.dll
AddType application/x-httpd-php .php
```

El lugar adecuado para añadir esas líneas es en el bloque de carga de módulos, que podemos encontrar si buscamos por el texto "LoadModule". Podemos añadir las líneas de carga del módulo PHP después de la carga de los otros módulos que vienen ya configurados en archivo httpd.conf de Apache.

Si no instalamos PHP en el directorio c:\php, debemos editar las líneas a colocar en el httpd.conf para colocar la ruta correcta al directorio donde está la librería php4apache2.dll.

### 2.3.5.- Un último paso

Antes de acabar y probar si PHP se ha instalado correctamente, necesitamos copiar una dll en el directorio sapi. Concretamente, la dll "php4ts.dll", que podemos encontrar en nuestro directorio de instalación de PHP es la que debemos copiar al directorio sapi, algo como "c:\phpsapi".

**Nota:** Esta acción no viene documentada en el manual de PHP, aunque sí no la llevamos a cabo no funcionará.

El error que obtenemos al tratar de arrancar el Apache es algo como:

```
Syntax error on line 173 of C:/Archivos de programa/Apache Group/Apache2/conf/httpd.conf:
Cannot load C:/php/sapi/php4apache2.dll into server: No se puede encontrar el módulo especificado.
```

Otra configuración que podemos aplicar al archivo httpd.conf es definir también como documento por defecto el archivo index.php en nuestro servidor Apache. El documento por defecto es generalmente index.html, pero lo habitual si vamos a programar con PHP es que también necesitemos definir index.php como documento a mostrar si no se indica otro documento del directorio al que se está accediendo.

El documento por defecto se define con la variable DirectoryIndex. Nos quedará una definición como esta:

```
DirectoryIndex index.html index.html.var index.php
```

### 2.3.6.- Probar si PHP está funcionando correctamente

Para terminar, podemos crear una página de prueba de PHP, que colocaremos en nuestro directorio de publicación de Apache, generalmente llamado htdocs, que se aloja dentro del directorio donde se ha instalado Apache, algo como "C:\Archivos de programa\Apache Group\Apache2\htdocs"

Podemos crear un archivo llamado, por ejemplo, "prueba.php", en el que colocaremos dentro el siguiente código:

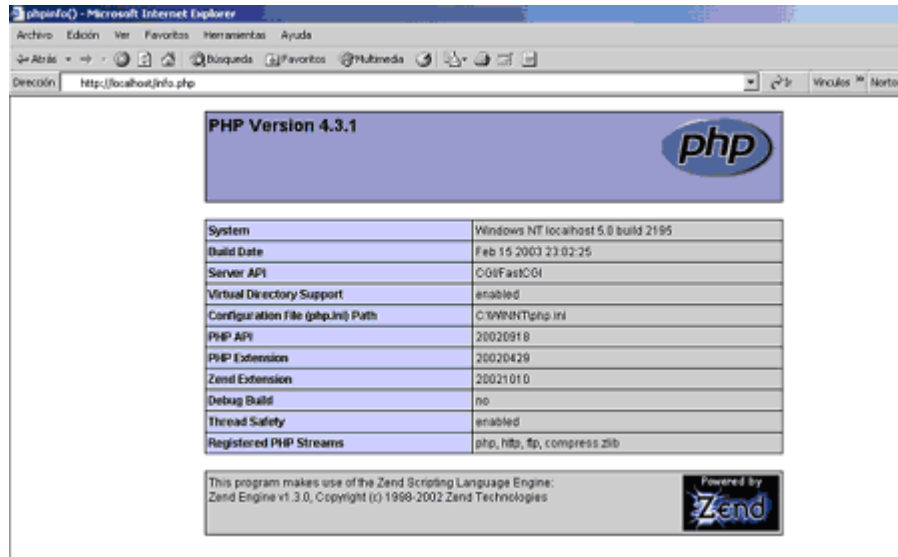
```
<?
phpinfo()
?>
```

Esta función simplemente creará una página de muestra de las configuraciones definidas para PHP en ese servidor.

Para acceder al archivo creado desde nuestro explorador, escribiremos en la barra de direcciones esta URL:

`http://localhost/prueba.php`

Debería aparecer un resultado como el de la siguiente imagen.



**Referencia:** En esta FAQ damos otras opciones para la instalación de PHP, utilizando programas que permiten instalar y configurar Apache + PHP + MySQL en un sencillo paso, accesible para todos los usuarios: [Cómo instalar PHP en Windows](#).

Artículo por Miguel Angel Alvarez

## 2.4.- Instalación de MySQL en Windows

*Pasos a seguir para la instalación de la base de datos MySQL y algunas ayudas básicas para comenzar con buen pie.*

Uno de los puntos fuertes de las páginas en PHP es la posibilidad de explotar bases de datos mediante funciones de una simplicidad y potencia muy agradecidas. Estas bases de datos pueden servir a nuestro sitio para almacenar contenidos de una forma sistemática que nos permita clasificarlos, buscarlos y editarlos rápida y fácilmente.

Una base de datos es sencillamente un conjunto de tablas en las que almacenamos distintos registros (artículos de una tienda virtual, proveedores o clientes de una empresa, películas en cartelera en el cine...). Estos registros son catalogados en función de distintos parámetros que los caracterizan y que presentan una utilidad a la hora de clasificarlos. Así, por ejemplo, los artículos de una tienda virtual podrían catalogarse a partir de distintos campos como puede ser un número de referencia, nombre del artículo, descripción, precio, proveedor...

La base de datos más difundida con el tandem UNIX-Apache es sin duda MySQL. Como para el caso de Apache, una versión para Windows está disponible y puede ser [descargada](#) gratis.

Su puesta a punto no entraña mucha dificultad. Una vez instalado el programa podemos ejecutar nuestras ordenes en modo MS-DOS. Para ello abrimos una ventana MS-DOS y nos colocamos en el directorio `bin` de `mysql`. En este directorio se encuentran los archivos ejecutables. Aquí habrá que encontrar un archivo llamado `mysqld`. En el caso de la versión más actual durante la redacción de este artículo este archivo es llamado `mysqld-shareware`. Una vez ejecutado este archivo podemos ejecutar el siguiente: `mysql`.

Llegados a este punto veremos cómo un mensaje de bienvenida aparece en nuestra pantalla. En estos momentos nos

encontramos dentro de la base de datos. A partir de ahí podemos realizar todo tipo de operaciones por sentencias SQL.

No vamos a entrar en una explicación pormenorizada del funcionamiento de esta base de datos ya que esto nos daría para un manual entero. Daremos como referencia nuestro [tutorial de SQL](#) a partir del cual se puede tener una idea muy práctica de las sentencias necesarias para la [creación](#) y edición de las tablas. También existe una documentación extensa en inglés en el directorio *Docs* de MySQL. A modo de resumen, aquí os proponemos además las operaciones más básicas que, combinadas nuestro [tutorial de SQL](#) pueden dar solución a gran parte de los casos que se os presenten:

Instrucción	Descripción
Show databases;	Muestra el conjunto de bases de datos presentes en el servidor
Use nombre_de_la_base	Determina la base de datos sobre la que vamos a trabajar
Create Database nombre_de_la_base;	Crea una nueva bd con el nombre especificado
Drop Database nombre_de_la_base;	Elimina la base de datos del nombre especificado
Show tables;	Muestra las tablas presentes en la base de datos actual
Describe nombre_de_la_tabla;	Describe los campos que componen la tabla
Drop Table nombre_de_la_tabla;	Borra la tabla de la base de datos
Load Data Local Infile "archivo.txt" Into Table nombre_de_la_tabla;	Crea los registros de la tabla a partir de un fichero de texto en el que separamos por tabulaciones todos los campos de un mismo registro.
Quit	Salir de MySQL

Para evitarnos el tener que editar nuestras tablas directamente sobre archivos de texto, puede resultar muy práctico usar cualquier otra base de datos con un editor y exportar a continuación la tabla en un archivo de texto configurado para dejar tabulaciones entre cada campo. Esto es posible en Access por ejemplo pinchando con el botón derecho sobre la tabla que queremos convertir y eligiendo la opción exportar. Una ventana de dialogo aparecerá en la que elegiremos guardar el archivo en tipo texto. El paso siguiente será elegir un formato delimitado por tabulaciones sin cualificador de texto.

Otra posibilidad que puede resultar muy práctica y que nos evita trabajar continuamente tecleando órdenes al estilo de antaño es servirse de programas en PHP o Perl ya existentes y descargables en la red. El más popular sin duda es

**phpMyAdmin.** Este tipo de scripts son ejecutados desde un navegador y pueden ser por tanto albergados en nuestro servidor o empleados en local para, a partir de ellos, administrar MySQL de una forma menos sufrida.

Asimismo, dentro del directorio bin de MySQL, podemos encontrar una pequeña aplicación llamada MySqlManager. Se trata de una interface windows, más agradable a la vista y al uso que la que obtenemos ejecutando el archivo *mysql*. En este caso, las sentencias SQL deben realizarse sin el punto y coma final.

*Artículo por Rubén Álvarez*

## 2.5.- Instalando PHP con IIS

*Cómo instalar paso a paso PHP con IIS.*

Descargar archivos de <http://www.php.net/downloads.php#v5>

Se recomienda bajar los archivos PHP 5.2.1 zip package y las extensiones adicionales PECL PECL 5.2.1 Win32 binaries

1. Crear directorio en C:\PHP y extraer los archivos de ph 5.21zip package y extraer los archivos de PECL 5.2.1 WIN32 binaries en el directorio C:\PHPEXT.

2. Renombrar el archivo php.ini-recommended como php.ini

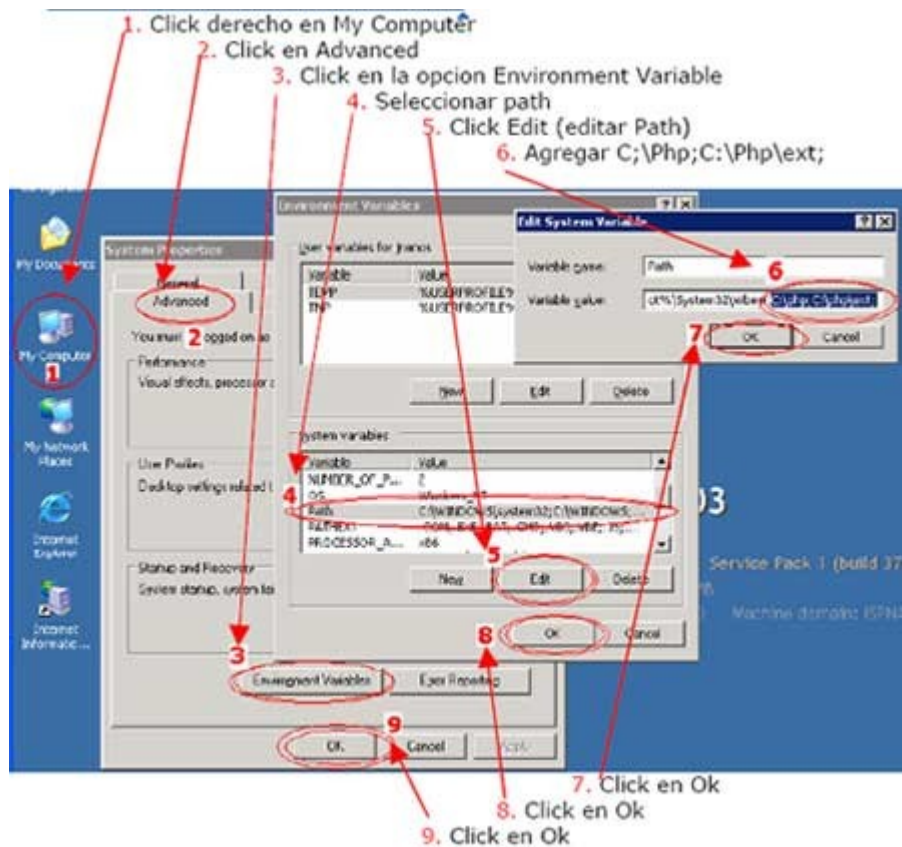
3. Modificar archivo php.ini en las siguientes variables

```
short_open_tag = On  
extension_dir = "c:\phpext"  
cgi.force_redirect = 0
```

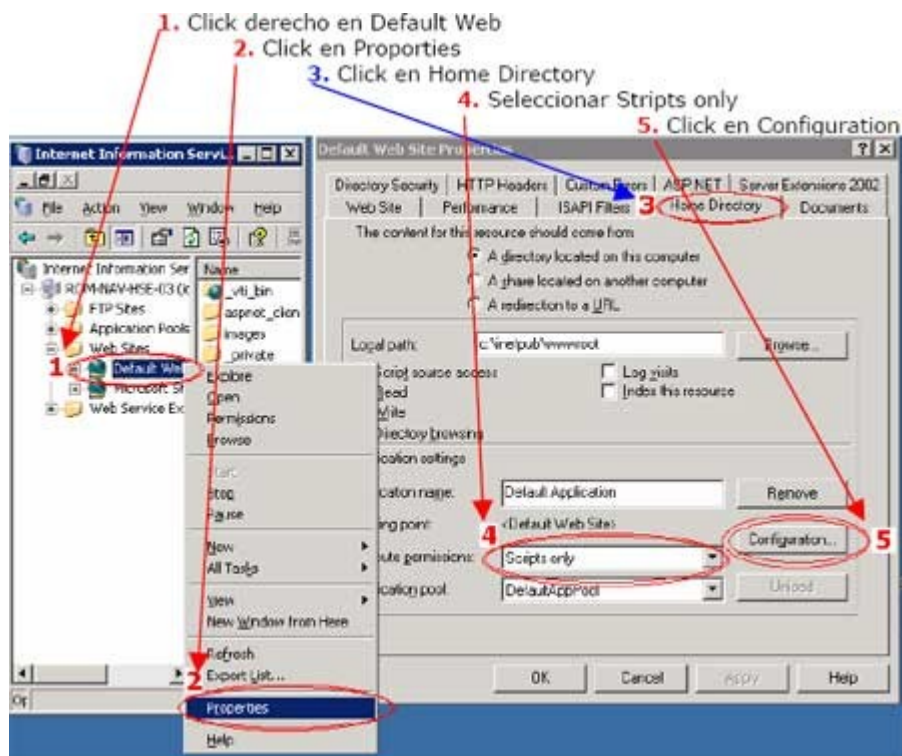
4. Copiar php.ini a c:\windows o c:\winnt

5. Registrar la DLL regsvr32 php5activescript.dll

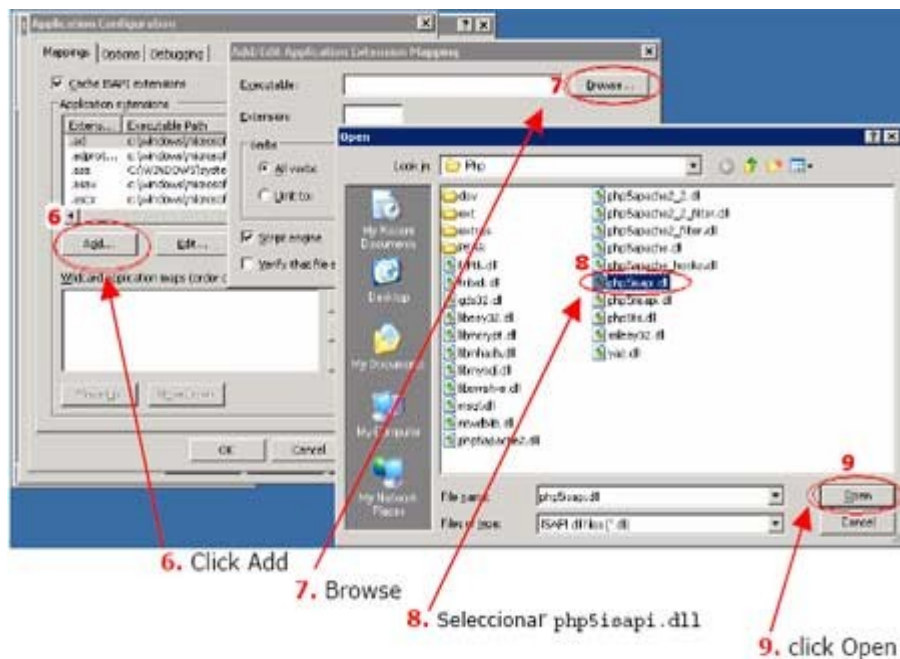
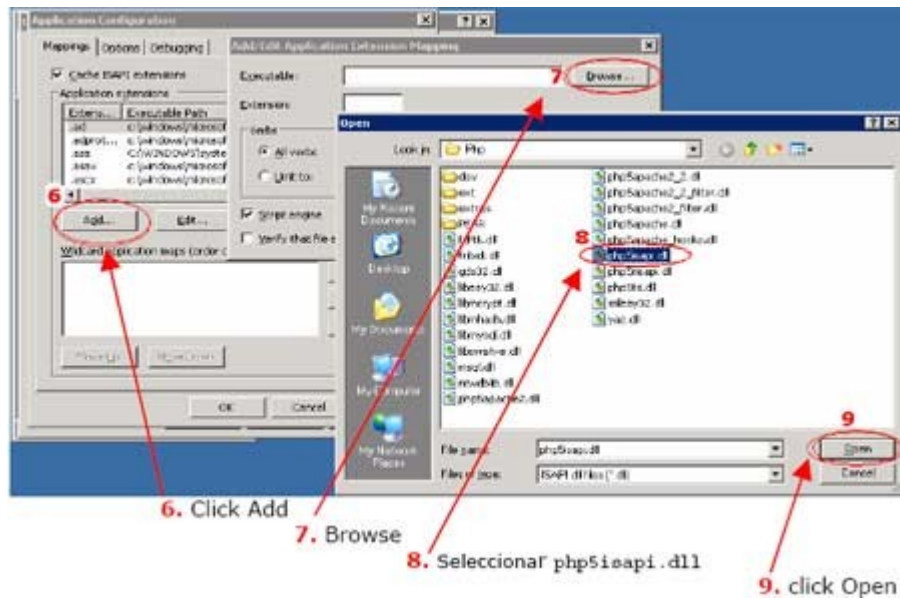
6. Agregar ruta de del directorio php y las extenciones en la opcion de Environment Variable de windows2003



## 7. Configuración en IIS

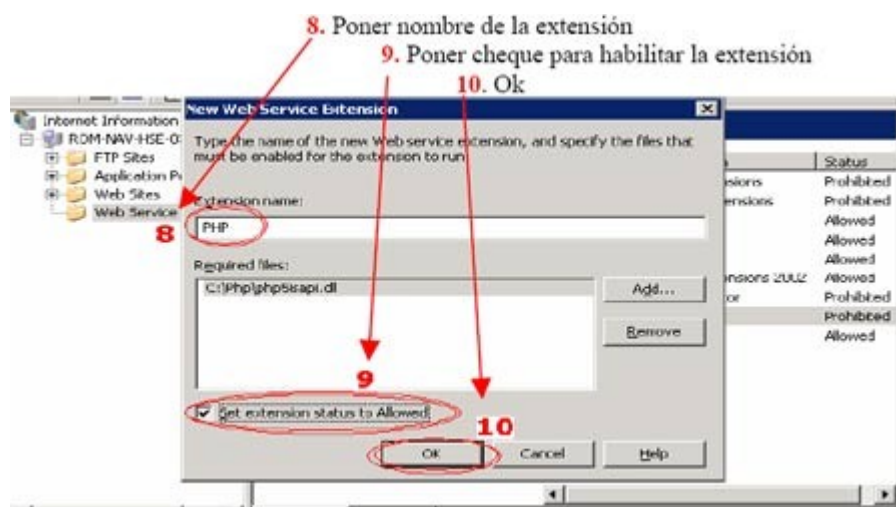
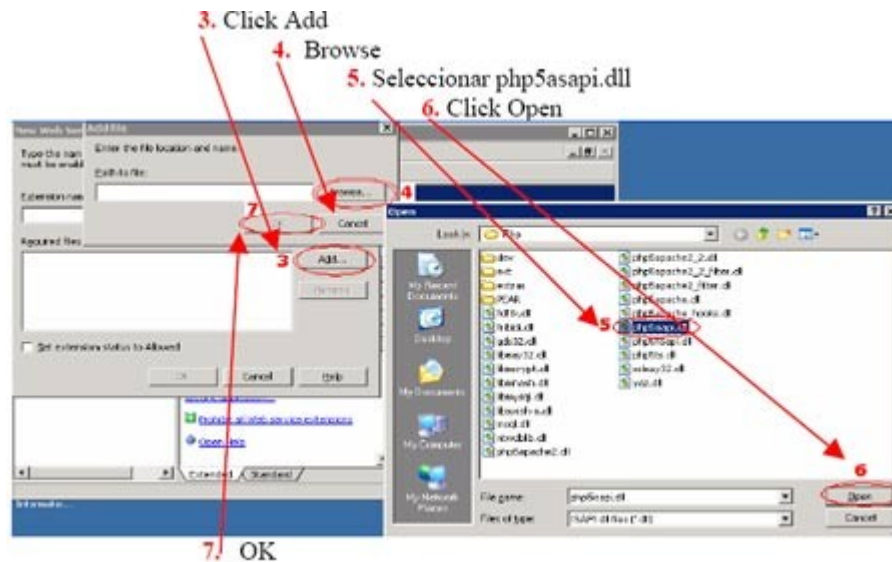
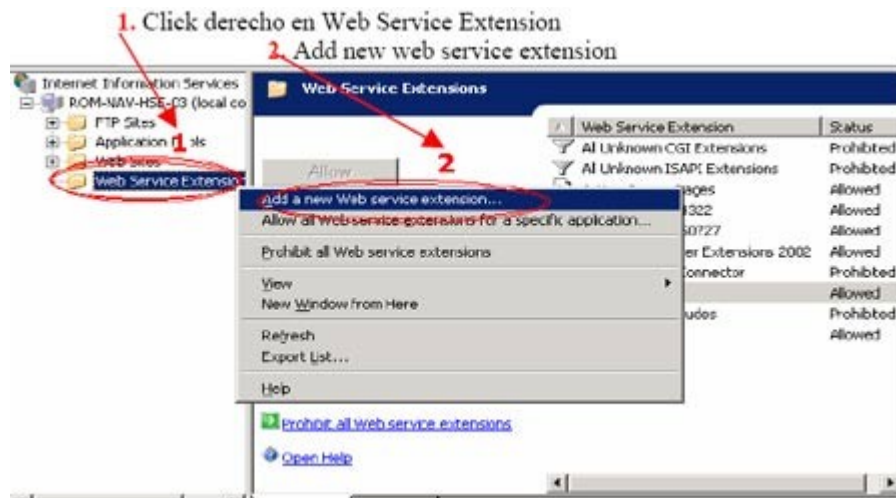




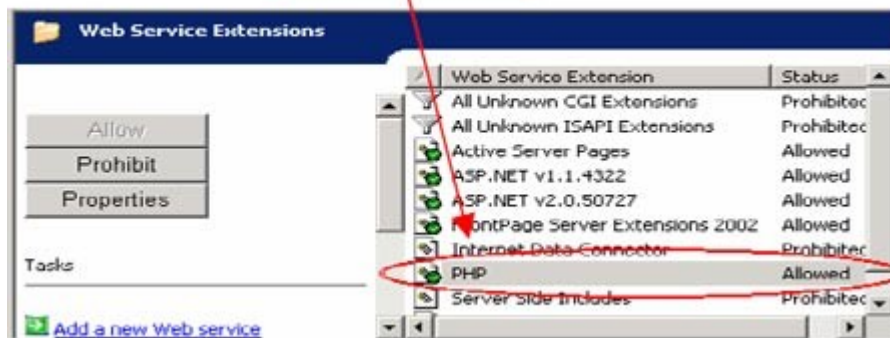


8. Configurando extensiones en IIS





### Extensión habilitada



#### 9. Probando PHP

Crear archivo index.php en C:\inetpub\wwwroot

Copiar el código siguiente en el archivo

```
<?php
// testing sessions
// check to see if files are being created
// in the session.save_path folder
session_start();
?>
<html>
<head>
<title>PHP Test</title>
</head>
<body>
<p>
The browser you're using is
<?php echo $_SERVER['HTTP_USER_AGENT']; ?>
</p>
<p>
<!-- test the browscap setup -->
Your browser's capabilities are: <br/>
<pre>
<?php print_r(get_browser(null, true)); ?>
</pre>
</p>
<?php phpinfo(); ?>
</body>
</html>
```

2. Abrir IE6 y poner la dirección <http://localhost/index.php>, si le da error verificar los pasos de la instalación.

Si no da error saldrá una pantalla con información sobre la versión de php.

Artículo por *Jorge Ramos*

## Parte 3:

# Primeros pasos con el lenguaje PHP

Empezamos a trabajar con el lenguaje de programación. En los siguientes capítulos del Manual de PHP explicaremos las generalidades sobre el lenguaje, como su sintaxis, las variables que podemos crear y sus tipos de datos, las variables del sistema que están disponibles sobre el servidor, operadores, etc.

### 3.1.- Introducción a la sintaxis PHP

*Explicamos las pautas principales a seguir para incluir PHP en el código de nuestra página, la forma de introducir comentarios.*

PHP se escribe dentro de la propia página web, junto con el código HTML y, como para cualquier otro tipo de lenguaje incluido en un código HTML, en PHP necesitamos especificar cuáles son las partes constitutivas del código escritas en este lenguaje. Esto se hace, como en otros casos, delimitando nuestro código por etiquetas. Podemos utilizar distintos modelos de etiquetas en función de nuestras preferencias y costumbres. Hay que tener sin embargo en cuenta que no necesariamente todas están configuradas inicialmente y que otras, como es el caso de `<% y %>` sólo están disponibles a partir de una determinada versión (3.0.4).

Estos modos de abrir y cerrar las etiquetas son:

```
<?           y           ?>
<%           y           %>
<?php       y           ?>
<script language="php">
```

Este último modo está principalmente aconsejado a aquellos que tengan el valor de trabajar con Front Page ya que, usando cualquier otro tipo de etiqueta, corremos el riesgo de que la aplicación nos la borre sin más debido a que se trata de un código incomprensible para ella.

El modo de funcionamiento de una página PHP, a grandes rasgos, no difiere del clásico para una página dinámica de lado servidor: El servidor va a reconocer la extensión correspondiente a la página PHP (phtml, php, php4,...) y antes de enviarla al navegador va a encargarse de interpretar y ejecutar todo aquello que se encuentre entre las etiquetas correspondientes al lenguaje PHP. El resto, lo enviara sin más ya que, asumirá que se trata de código HTML absolutamente comprensible por el navegador.

Otra característica general de los scripts en PHP es la forma de separar las distintas instrucciones. Para hacerlo, hay que acabar cada instrucción con un punto y coma ";". Para la última expresión, la que va antes del cierre de etiqueta, este formalismo no es necesario.

Incluimos también en este capítulo la sintaxis de comentarios. Un comentario, para aquellos que no lo sepan, es una frase o palabra que nosotros incluimos en el código para comprenderlo más fácilmente al volverlo a leer un tiempo después y que, por supuesto, el ordenador tiene que ignorar ya que no va dirigido a él sino a nosotros mismos. Los comentarios tienen una gran utilidad ya que es muy fácil olvidarse del funcionamiento de un script programado un tiempo atrás y resulta muy útil si queremos hacer rápidamente comprensible nuestro código a otra persona.

Pues bien, la forma de incluir estos comentarios es variable dependiendo si queremos escribir una línea o más. Veamos esto con un primer ejemplo de script:

```
<?
$mensaje="Tengo hambre!!"; //Comentario de una línea
echo $mensaje; #Este comentario también es de una línea
/*En este caso
mi comentario ocupa
varias líneas, lo ves? */
?>
```

#### Ejecutar script

Si usamos doble barra (//) o el símbolo # podemos introducir comentarios de una línea. Mediante /\* y \*/ creamos comentarios multilínea. Por supuesto, nada nos impide de usar estos últimos en una sola línea.

No os preocupéis si no comprendéis el texto entre las etiquetas, todo llegará. Os adelantamos que las variables en PHP se definen anteponiendo un símbolo de dólar (\$) y que la instrucción *echo* sirve para sacar en pantalla lo que hay escrito a continuación.

Recordamos que todo el texto insertado en forma de comentario es completamente ignorado por el servidor. Resulta importante acostumbrarse a dejar comentarios, es algo que se agradece con el tiempo.

Artículo por *Rubén Álvarez*

## 3.2.- Variables en PHP

*Tipos de variables, características generales y aspectos específicos de PHP de estos elementos básicos de la programación.*

Las variables son uno de los primeros temas que tenemos que conocer en PHP y en la mayoría de los lenguajes de programación. Así que a continuación vamos a tratar este tema dentro del [Manual de PHP](#), explicando los tipos de variables que podremos encontrar.

Anteriormente en DesarrolloWeb.com, en el manual de páginas dinámicas, ya habíamos introducido el [concepto de variable](#). En el capítulo anterior comentábamos que, para PHP, las variables eran definidas anteponiendo el símbolo dólar (\$) al nombre de la variable que estábamos definiendo.

Dependiendo de la información que contenga, una variable puede ser considerada de uno u otro tipo:

Variables numéricas Almacenan cifras		
Enteros	\$entero=2002;	Numeros sin decimales
Real	\$real=3.14159;	Numeros con o sin decimal
Variables alfanuméricas Almacenan textos compuestos de números y/o cifras		
Cadenas	Almacenan variables alfanuméricas	\$cadena="Hola amigo";

Tablas		
Almacenan series de informaciones numéricas y/o alfanuméricas		
Arrays	Son las variables que guardan las tablas	<code>\$sentido[1]="ver"; \$sentido[2]="tocar"; \$sentido[3]="oir"; \$sentido[4]="gusto"; \$sentido[5]="oler";</code>
Objetos		
Se trata de conjuntos de variables y funciones asociadas. Presentan una complejidad mayor que las variables vistas hasta ahora pero su utilidad es más que interesante.		

A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables. En efecto, cuando definimos una variable asignándole un valor, el ordenador le atribuye un tipo. Si por ejemplo definimos una variable entre comillas, la variable será considerada de tipo cadena:

```
$variable="5"; //esto es una cadena
```

Sin embargo si pedimos en nuestro script realizar una operación matemática con esta variable, no obtendremos un mensaje de error sino que la variable cadena será asimilada a numérica:

```
<?
$cadena="5"; //esto es una cadena
$entero=3; //esto es un entero
echo $cadena+$entero
?>
```

### Ejecutar script

Este script dará como resultado "8". La variable cadena ha sido asimilada en entero (aunque su tipo sigue siendo cadena) para poder realizar la operación matemática. Del mismo modo, podemos operar entre variables tipo entero y real. No debemos preocuparnos de nada, PHP se encarga durante la ejecución de interpretar el tipo de variable necesario para el buen funcionamiento del programa.

Sin embargo, en contraste, hay que tener cuidado en no cambiar mayúsculas por minúsculas ya que, en este sentido, PHP es sensible. Conviene por lo tanto trabajar ya sea siempre en mayúsculas o siempre en minúsculas para evitar este tipo de malentendidos a veces muy difíciles de localizar.

### **Variables asignadas por referencia**

En PHP también podemos asignar variables por referencia. En ese caso no se les asigna un valor, sino otra variable, de tal modo que las dos variables comparten espacio en memoria para el mismo dato.

La notación para asignar por referencia es colocar un "&" antes del nombre de la variable.

```
<?php
$foo = 'Bob'; // Asigna el valor 'Bob' a $foo
$bar = &$foo; // Referencia $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $foo; // $foo también se modifica.
echo $bar;
?>
```

Esto dará como resultado la visualización dos veces del string "Mi nombre es Bob". Algo como:

Mi nombre es BobMi nombre es Bob

**Nota:** Tenemos un [videotutorial que nos habla de las variables PHP](#)

Artículo por *Rubén Álvarez*

## 3.3.- Cambio del tipo de las variables en PHP

*Formas en que una variable de PHP puede ver variado su tipo.*

PHP no requiere que indiquemos el tipo que va a contener una variable, sino que lo deduce del valor que asignemos a la variable. Asimismo, se encarga de actualizar automáticamente el tipo de la variable cada vez que le asignamos un nuevo valor.

Por ello, para cambiar el tipo de una variable simplemente le asignamos un valor con un nuevo tipo.

**Nota:** Se excluyen en este caso el cambio de variables a tipo Array porque la sintaxis puede resultar ambigua al expresar ese código, es decir, puede darse el caso de que una línea de código pueda significar dos cosas.

```
$a = "1";  
// $a es una cadena  
$a[0] = "f";  
// ¿Estamos editando el índice de la cadena o forzando a array?
```

### 3.3.1.- Forzado

En cualquier caso, podemos forzar una variable para que cambie de tipo con la función `setType()`.

```
setType($variable,"nuevo_tipo");
```

la función `setType()` actualiza el tipo de `$variable` a `"nuevo_tipo"` y devuelve un booleano indicando si hubo éxito o no en la conversión.

Entre `"nuevo_tipo"` tenemos:

- `"integer"`
- `"double"`
- `"string"`
- `"array"`
- `"object"`

También podemos hacer que una variable se comporte como un tipo determinado forzándola, de la misma manera a como se hace en el lenguaje C.

```
$variable = "23";  
$variable = (int) $variable;
```

Los forzados permitidos son:

- `(int)`, `(integer)` - fuerza a entero (`integer`)
- `(real)`, `(double)`, `(float)` - fuerza a doble (`double`)
- `(string)` - fuerza a cadena (`string`)

· (array) - fuerza a array (array)  
(object) - fuerza a objeto (object)

Artículo por *Miguel Angel Alvarez*

### 3.4.- Variables de sistema en PHP

*Qué son y para qué sirven estas variables del servidor. Comentamos algunas de las más útiles.*

Dada su naturaleza de lenguaje de lado servidor, PHP es capaz de darnos acceso a toda una serie de variables que nos informan sobre nuestro servidor y sobre el cliente. La información de estas variables es atribuida por el servidor y en ningún caso nos es posible modificar sus valores directamente mediante el script. Para hacerlo es necesario influir directamente sobre la propiedad que definen.

Existen multitud de variables de este tipo, algunas sin utilidad aparente y otras realmente interesantes y con una aplicación directa para nuestro sitio web. Aquí os enumeramos algunas de estas variables y la información que nos aportan:

Variable	Descripción
\$HTTP_USER_AGENT	Nos informa principalmente sobre el sistema operativo y tipo y versión de navegador utilizado por el internauta. Su principal utilidad radica en que, a partir de esta información, podemos redireccionar nuestros usuarios hacia páginas optimizadas para su navegador o realizar cualquier otro tipo de acción en el contexto de un navegador determinado.
\$HTTP_ACCEPT_LANGUAGE	Nos devuelve la o las abreviaciones de la lengua considerada como principal por el navegador. Esta lengua o lenguas principales pueden ser elegidas en el menú de opciones del navegador. Esta variable resulta también extremadamente útil para enviar al internauta a las páginas escritas en su lengua, si es que existen.
\$HTTP_REFERER	Nos indica la URL desde la cual el internauta ha tenido acceso a la página. Muy interesante para generar botones de "Atrás" dinámicos o para crear nuestros propios sistemas de estadísticas de visitas.
\$PHP_SELF	Nos devuelve una cadena con la URL del script que está siendo ejecutado. Muy interesante para crear botones para recargar la página.
\$HTTP_GET_VARS	Se trata de un array que almacena los nombres y contenidos de las variables enviadas al script por URL o



	por formularios GET
\$HTTP_POST_VARS	Se trata de un array que almacena los nombres y contenidos de las variables enviadas al script por medio de un formulario POST
\$HTTP_COOKIE_VARS	Se trata de un array que almacena los nombres y contenidos de las cookies. Veremos qué son más adelante.
\$PHP_AUTH_USER	Almacena la variable usuario cuando se efectúa la entrada a páginas de acceso restringido. Combinado con \$PHP_AUTH_PW resulta ideal para controlar el acceso a las páginas internas del sitio.
\$PHP_AUTH_PW	Almacena la variable password cuando se efectúa la entrada a páginas de acceso restringido. Combinado con \$PHP_AUTH_USER resulta ideal para controlar el acceso a las páginas internas del sitio.
\$REMOTE_ADDR	Muestra la dirección IP del visitante.
\$DOCUMENT_ROOT	Nos devuelve el path físico en el que se encuentra alojada la página en el servidor.
\$PHPSESSID	Guarda el identificador de sesión del usuario. Veremos más adelante en qué consisten las sesiones.

No todas estas variables están disponibles en la totalidad de servidores o en determinadas versiones de un mismo servidor. además, algunas de ellas han de ser previamente activadas o definidas por medio de algún acontecimiento. Así, por ejemplo, la variable \$HTTP\_REFERER no estará definida a menos que el internauta acceda al script a partir de un enlace desde otra página.

### 3.4.1.- Variables superglobales

A partir de PHP 4.1.0, se dispone de un conjunto de variables de tipo array que mantienen información del sistema, llamadas superglobales porque se definen automáticamente en un ámbito global.

Estas variables hacen referencia a las mismas que se accedían antes por medio de los arrays del tipo \$HTTP\_\*\_VARS. Éstas todavía existen, aunque a partir de PHP 5.0.0 se pueden desactivar con la directiva `register_long_arrays`.

La lista de estas variables, extraída directamente de la documentación de PHP es la siguiente:

### 3.4.2.- \$GLOBALS

Contiene una referencia a cada variable disponible en el espectro de las variables del script. Las llaves de esta matriz son los nombres de las variables globales. \$GLOBALS existe desde PHP 3.

**\$\_SERVER**

Variables definidas por el servidor web ó directamente relacionadas con el entorno en don el script se esta ejecutando. Análoga a la antigua matriz \$HTTP\_SERVER\_VARS (la cual está todavía disponible, aunque no se use).

**\$\_GET**

Variables proporcionadas al script por medio de HTTP GET. Análoga a la antigua matriz \$HTTP\_GET\_VARS (la cual está todavía disponible, aunque no se use).

**\$\_POST**

Variables proporcionadas al script por medio de HTTP POST. Análoga a la antigua matriz \$HTTP\_POST\_VARS (la cual está todavía disponible, aunque no se use).

**\$\_COOKIE**

Variables proporcionadas al script por medio de HTTP cookies. Análoga a la antigua matriz \$HTTP\_COOKIE\_VARS (la cual está todavía disponible, aunque no se use).

**\$\_FILES**

Variables proporcionadas al script por medio de la subida de ficheros via HTTP . Análoga a la antigua matriz \$HTTP\_POST\_FILES (la cual está todavía disponible, aunque no se use). Vea también Subiendo ficheros por método POST para más información.

**\$\_ENV**

Variables proporcionadas al script por medio del entorno. Análoga a la antigua matriz \$HTTP\_ENV\_VARS (la cual está todavía disponible, aunque no se use).

**\$\_REQUEST**

Variables proporcionadas al script por medio de cualquier mecanismo de entrada del usuario y por lo tanto no se puede confiar en ellas. La presencia y el orden en que aparecen las variables en esta matriz es definido por la directiva de configuración variables\_order. Esta matriz no tiene un análogo en versiones anteriores a PHP 4.1.0. Vea también `import_request_variables()`.

**\$\_SESSION**

Variables registradas en la sesión del script. Análoga a la antigua matriz \$HTTP\_SESSION\_VARS (la cual está todavía disponible, aunque no se use). Vea también la sección Funciones para el manejo de sesiones para más información.

*Artículo por Rubén Álvarez*

## 3.5.- Ámbito de las variables en PHP

*Explicamos con detalle el ámbito de existencia de una variable en PHP y distinguimos entre variables globales y locales.*

En cualquier lenguaje de programación las variables tienen un ámbito, que es el lugar o lugares donde tienen validez. El ámbito varía en función de donde se hayan creado esas variables, pudiendo ser globales o locales.

En PHP, todas las variables creadas en la página, fuera de funciones, son variables globales a la página. Por su parte, las variables creadas dentro de una función son variables locales a esa función.

Las variables globales se pueden acceder en cualquier lugar de la página, mientras que las variables locales sólo tienen validez dentro de la función donde han sido creadas. De modo que una variable global la podemos acceder dentro de cualquier parte del código, mientras que si intentamos acceder a una variable local fuera de la función donde fue creada, nos encontraremos con que esa variable no tiene contenido alguno.

Ahora bien, si intentamos acceder a una variable global dentro de una función, en principio también nos encontraremos con

que no se tiene acceso a su valor. Esto es así en PHP por motivos de claridad del código, para evitar que se pueda prestar a confusión el hecho de usar dentro de una función una variable que no ha sido declarada por ningún sitio cercano.

**Nota:** tal vez resulten desconocidos los conceptos sobre funciones, que se tratan más adelante en este manual: [funciones en PHP](#)

Entonces, si queremos utilizar una variable global a la página dentro de una función, tenemos que especificar de alguna manera que esa variable que vamos a utilizar es una global. Existen en PHP un par de maneras de utilizar variables globales a la página dentro de una función. Son las siguientes:

### Matriz GLOBALS

Existe un array en PHP llamado \$GLOBALS, que guarda una referencia a todas las variables creadas de manera global a la página. Es una matriz o array asociativo, de los que en lugar de índices numéricos utilizan índices de texto, donde cada índice es el nombre que hemos dado a la variable y cada valor es el contenido de cada variable.

Supongamos que tenemos esta declaración de variables globales a la página, es decir, fuera de cualquier función:

```
$mivariable = "pepe";  
$otravariable = 1234;
```

Si queremos acceder a esas variables dentro de una función utilizando el array \$GLOBALS tendríamos este código:

```
function mifuncion(){  
    //estoy dentro de la función, para acceder a las variables utilizo $GLOBALS  
    echo $GLOBALS["mivariable"];  
    echo $GLOBALS["otravariable"];  
}
```

Como se puede ver, se accede al contenido de las variables globales con el array \$GLOBALS, utilizando como índices de la matriz los nombres de variables que deseamos mostrar.

Esto imprimiría por pantalla el texto "pepe1234", el valor de las dos variables uno detrás del otro.

### Declaración de uso de variables globales dentro de una función

Otra cosa que podemos hacer para acceder a variables globales dentro de una función es especificar al comienzo de dicha función la lista de variables que vamos a utilizar dentro. Para especificar esas variables utilizamos la palabra "global" seguida de la lista de variables que se van a utilizar del entorno global.

```
function mifuncion(){  
    global $mivariable, $otravariable  
    //con esa línea dentro de la función, declaramos el uso de variables globales  
    echo $mivariable;  
    echo $otravariable;  
}
```

Como vemos, con "global" se especifica que vamos a utilizar unas variables que fueron declaradas como globales a la página. Una vez hecho esto, ya podemos acceder a esas variables globales como si estuvieran declaradas dentro de la función.

Cualquier alteración que hagamos a las variables dentro de la función permanecerá cuando se haya salido de la función, tanto si accedemos a través del array \$GLOBALS o declarando con "global" el uso de esas variables.

*Artículo por Miguel Ángel Álvarez*

## 3.6.- Operadores

### *Lista descriptiva de los operadores más frecuentemente utilizados*

Las variables, como base de información de un lenguaje, pueden ser creadas, modificadas y comparadas con otras por medio de los llamados operadores. En los capítulos anteriores hemos utilizado en nuestros ejemplos algunos de ellos.

En este capítulo pretendemos listar los más importantes y así dar constancia de ellos para futuros ejemplos.

### 3.6.1.- Operadores aritméticos

Nos permiten realizar operaciones numéricas con nuestras variables

+	Suma
-	Resta
*	Multiplicación
/	División
%	Devuelve el resto de la división

**Referencia:** El operador aritmético que puede resultar más desconocido para los lectores es el operador %. Explicamos con mayor detenimiento su funcionamiento y un ejemplo en el que es útil en el taller: [Listas de elementos con colores alternos en PHP](#).

### 3.6.2.- Operadores de comparación

Se utilizan principalmente en nuestras condiciones para comparar dos variables y verificar si cumple o no la propiedad del operador.

==	Igualdad
!=	Desigual
<	Menor que
<=	Menor igual que
>	Mayor que
>=	Mayor igual que

### 3.6.3.- Operadores lógicos

Se usan en combinación con los operadores de comparación cuando la expresión de la condición lo requiere.

And	Y
Or	O

!	No

### 3.6.4.- Operadores de incremento

Sirven para aumentar o disminuir de una unidad el valor de una variable

++\$variable	Aumenta de 1 el valor de \$variable
--\$variable	Reduce de uno el valor de \$variable

### 3.6.5.- Operadores combinados

Una forma habitual de modificar el valor de las variables es mediante los operadores combinados:

\$variable += 10	Suma 10 a \$variable
\$variable -= 10	Resta 10 a \$variable
\$variable .= "añado"	Concatena las cadenas \$variable y "añado"

Este tipo de expresiones no son más que abreviaciones de otras formas más clásicas:

\$variable += 10

es lo mismo que:

\$variable = \$variable+10

*Artículo por Rubén Álvarez*

## Parte 4:

# Arrays y cadenas

Nos detenemos para ver cómo son los arrays en PHP, la estructura de datos más esencial que existe. Además aprenderemos a trabajar con cadenas en PHP.

### 4.1.- Tablas o Arrays en PHP

*Creación de tablas por medio de variables tipo array. Utilidad y funciones útiles relacionadas.*

Un tipo de variable que ya hemos descrito pero puede ser relativamente complicado a asimilar con respecto a la mayoría son los arrays. Un array es una variable que está compuesta de varios elementos cada uno de ellos catalogado dentro de ella misma por medio de una clave.

En el capítulos anteriores poníamos el ejemplo de un array llamado sentido que contenía los distintos sentidos del ser humano:

```
$sentido[1]="ver";
$sentido[2]="tocar";
$sentido[3]="oir";
$sentido[4]="gustar";
$sentido[5]="oler";
```

En este caso este array cataloga sus elementos, comunmente llamados valores, por números. Los números del 1 al 5 son por lo tanto las claves y los sentidos son los valores asociados. Nada nos impide emplear nombres (cadenas) para clasificarlos. Lo único que deberemos hacer es entrecomillarlos:

```
<?
$moneda["espana"]="Peseta";
$moneda["francia"]="Franco";
$moneda["usa"]="Dolar";
?>
```

Otra forma de definir idénticamente este mismo array y que nos puede ayudar para la creación de arrays más complejos es la siguiente sintaxis:

```
<?
$moneda=array("espana"=> "Peseta","francia" => "Franco","usa" => "Dolar");
?>
```

Una forma muy practica de almacenar datos es mediante la creación de arrays multidimensionales (tablas). Pongamos el ejemplo siguiente: Queremos almacenar dentro de una misma tabla el nombre, moneda y lengua hablada en cada país. Para hacerlo podemos emplear un array llamado país que vendrá definido por estas tres características (claves). Para crearlo, deberíamos escribir una expresión del mismo tipo que la vista anteriormente en la que meteremos una array dentro del otro. Este proceso de incluir una instruccion dentro de otra se llama anidar y es muy corriente en programación:

```
<?
$pais=array
(
    "espana" =>array
    (
        "nombre"=>"España",
        "lengua"=>"Castellano",
        "moneda"=>"Peseta"
    ),
```

```
"francia" =>array
(
    "nombre"=>"Francia",
    "lengua"=>"Francés",
    "moneda"=>"Franco"
)
);
echo $pais["espana"]["moneda"] //Saca en pantalla: "Peseta"
?>
```

### Ejecutar script

Antes de entrar en el detalle de este pequeño script, comentemos algunos puntos referentes a la sintaxis. Como puede verse, en esta secuencia de script, no hemos introducido punto y coma ";" al final de cada línea. Esto es simplemente debido a que lo que hemos escrito puede ser considerado como una sola instrucción. En realidad, somos nosotros quienes decidimos cortarla en varias líneas para, así, facilitar su lectura. La verdadera instrucción acabaría una vez definido completamente el array y es precisamente ahí donde hemos colocado el único punto y coma. Por otra parte, podéis observar cómo hemos jugado con el tabulador para separar unas líneas más que otras del principio. Esto también lo hacemos por cuestiones de claridad, ya que nos permite ver qué partes del código están incluidas dentro de otras. Es importante acostumbrarse a escribir de esta forma del mismo modo que a introducir los comentarios ya que la claridad de los scripts es fundamental a la hora de depurarlos. Un poco de esfuerzo a la hora de crearlos puede ahorrarnos muchas horas a la hora de corregirlos o modificarlos meses más tarde.

Pasando ya al comentario del programa, como podéis ver, éste nos permite almacenar tablas y, a partir de una simple petición, visualizarlas un determinado valor en pantalla.

Lo que es interesante es que la utilidad de los arrays no acaba aquí, sino que también podemos utilizar toda una serie de funciones creadas para ordenarlos por orden alfabético directo o inverso, por claves, contar el número de elementos que componen el array además de poder movernos por dentro de él hacia delante o atrás.

Muchas son las [funciones](#) propuestas por PHP para el tratamiento de arrays, no vamos a entrar aquí en una descripción de las mismas. Sólo incluiremos esta pequeña tabla que puede ser complementada, si necesario, con la [documentación](#) que ya hemos mencionado.

Función	Descripción
array_values(mi_array)	Lista los valores contenidos en mi_array
asort(mi_array) y arsort(mi_array)	Ordena por orden alfabético directo o inverso en función de los valores
count(mi_array)	Nos da el número de elementos de nuestro array
ksort(mi_array) y krsort(mi_array)	Ordena por orden alfabético directo o inverso en función de las claves
list(\$variable1, \$variable2...)=mi_array	Asigna cada una variable a cada uno de los valores del array
next(mi_array), prev(mi_array), reset(mi_array) y end(mi_array)	Nos permiten movernos por dentro del array con un puntero hacia delante, atrás y al principio y al final.
each(mi_array)	Nos da el valor y la clave del elemento en el que nos



	encontramos y mueve al puntero al siguiente elemento.
--	---

De gran utilidad es también el bucle [foreach](#) que recorre de forma secuencial el array de principio a fin.

Para complementar esta información resultará de gran interés el artículo [Trabajo con tablas o arrays en PHP](#) y para los que prefieran la formación en vídeo, recomendamos ver los [videotutoriales sobre los arrays en PHP](#).

Artículo por *Rubén Álvarez*

## 4.2.- Trabajo con tablas o arrays en PHP

*Vemos algunas de las funciones típicas del trabajo con arrays a través de una pequeña explicación y un ejemplo de uso.*

Vamos a ver varios ejemplos de trabajo con arrays (arreglos, vectores, matrices o tablas en castellano) en PHP que ilustrarán un poco el funcionamiento de algunas de las funciones de arrays más populares que trae consigo PHP.

Sin más, vamos a introducirnos en materia con varios ejemplos interesantes de manejo de vectores.

**Referencia:** Los arrays en PHP se explican en el artículo [Tablas o Arrays en PHP](#).

### 4.2.1.- Modificar el número de elementos de un array

Ahora vamos a ver varios ejemplos mediante los cuales nuestros arrays pueden aumentar o reducir el número de casillas disponibles.

#### Reducir el tamaño de un array

##### **array\_slice()**

Para disminuir el número de casillas de un arreglo tenemos varias funciones. Entre ellas, `array_slice()` la utilizamos cuando queramos recortar algunas casillas del arreglo, sabiendo los índices de las casillas que deseamos conservar.

Recibe tres parámetros. El array, el índice del primer elemento y el número de elementos a tomar, siendo este último parámetro opcional.

En el ejemplo siguiente tenemos un array con cuatro nombres propios. En la primera ejecución de `array_slice()` estamos indicando que deseamos tomar todos los elementos desde el índice 0 (el principio) hasta un número total de 3 elementos.

El segundo `array_slice()` indica que se tomen todos los elementos a partir del índice 1 (segunda casilla).

```
<?
$entrada = array ("Miguel", "Pepe", "Juan", "Julio", "Pablo");

//modifico el tamaño
$salida = array_slice ($entrada, 0, 3);
//muestro el array
foreach ($salida as $actual)
    echo $actual . "<br>";

echo "<p>";

//modifico otra vez
$salida = array_slice ($salida, 1);
//muestro el array
foreach ($salida as $actual)
    echo $actual . "<br>";
?>
```

Tendrá como salida:

Miguel  
Pepe  
Juan

Pepe  
Juan

### **array\_shift()**

Esta función extrae el primer elemento del array y lo devuelve. Además, acorta la longitud del array eliminando el elemento que estaba en la primera casilla. Siempre hace lo mismo, por tanto, no recibirá más que el array al que se desea eliminar la primera posición.

En el código siguiente se tiene el mismo vector con nombres propios y se ejecuta dos veces la función `array_shift()` eliminando un elemento en cada ocasión. Se imprimen los valores que devuelve la función y los elementos del array resultante de eliminar la primera casilla.

```
<?
$entrada = array ("Miguel", "Pepe", "Juan", "Julio", "Pablo");

//quito la primera casilla
$salida = array_shift ($entrada);
//muestro el array
echo "La función devuelve: " . $salida . "<br>";
foreach ($entrada as $actual)
    echo $actual . "<br>";

echo "<p>";

//quito la primera casilla, que ahora sería la segunda del array original
$salida = array_shift ($entrada);
echo "La función devuelve: " . $salida . "<br>";
//muestro el array
foreach ($entrada as $actual)
    echo $actual . "<br>";
?>
```

Da como resultado:

La función devuelve: Miguel  
Pepe  
Juan  
Julio  
Pablo

La función devuelve: Pepe  
Juan  
Julio  
Pablo

### **unset()**

Se utiliza para destruir una variable dada. En el caso de los arreglos, se puede utilizar para eliminar una casilla de un array asociativo (los que no tienen índices numéricos sino que su índice es una cadena de caracteres).

Veamos el siguiente código para conocer cómo definir un array asociativo y eliminar luego una de sus casillas.

```
<?
$estadios_futbol = array("Barcelona"=> "Nou Camp","Real Madrid" => "Santiago Bernabeu","Valencia" =>
"Mestalla","Real Sociedad" => "Anoeta");

//mostramos los estadios
foreach ($estadios_futbol as $indice=>$actual)
    echo $indice . " -- " . $actual . "<br>";

echo "<p>";

//eliminamos el estadio asociado al real madrid
unset ($estadios_futbol["Real Madrid"]);
```

```
//mostramos los estadios otra vez
foreach ($estadios_futbol as $indice=>$actual)
echo $indice . " -- " . $actual . "<br>";
?>
```

La salida será la siguiente:

```
Barcelona -- Nou Camp
Real Madrid -- Santiago Bernabeu
Valencia -- Mestalla
Real Sociedad -- Anoeta

Barcelona -- Nou Camp
Valencia -- Mestalla
Real Sociedad -- Anoeta
```

### 4.2.2.- Aumentar el tamaño de un array

Tenemos también a nuestra disposición varias funciones que nos pueden ayudar a aumentar el número de casillas de un arreglo.

#### **array\_push()**

Inserta al final del array una serie de casillas que se le indiquen por parámetro. Por tanto, el número de casillas del array aumentará en tantos elementos como se hayan indicado en el parámetro de la función. Devuelve el número de casillas del array resultante.

Veamos este código donde se crea un arreglo y se añaden luego tres nuevos valores.

```
<?
$tabla = array ("Lagartija", "Araña", "Perro", "Gato", "Ratón");

//aumentamos el tamaño del array
array_push($tabla, "Gorrión", "Paloma", "Oso");

foreach ($tabla as $actual)
    echo $actual . "<br>";
?>
```

Da como resultado esta salida:

```
Lagartija
Araña
Perro
Gato
Ratón
Gorrión
Paloma
Oso
```

#### **array\_merge()**

Ahora vamos a ver cómo unir dos arrays utilizando la función `array_merge()`. A ésta se le pasan dos o más arrays por parámetro y devuelve un arreglo con todos los campos de los vectores pasados.

En este código de ejemplo creamos tres arrays y luego los unimos con la función `array_merge()`

```
<?
$tabla = array ("Lagartija", "Araña", "Perro", "Gato", "Ratón");
$tabla2 = array ("12", "34", "45", "52", "12");
$tabla3 = array ("Sauce", "Pino", "Naranja", "Chopo", "Perro", "34");

//aumentamos el tamaño del array
$resultado = array_merge($tabla, $tabla2, $tabla3);

foreach ($resultado as $actual)
    echo $actual . "<br>";
?>
```

Da como resultado:

Lagartija  
Araña  
Perro  
Gato  
Ratón  
12  
34  
45  
52  
12  
Sauce  
Pino  
Naranja  
Chopo  
Perro  
34

Una última cosa. También pueden introducirse nuevas casillas en un arreglo por los métodos habituales de asignar las nuevas posiciones en el array a las casillas que necesitemos.

En arrays normales se haría así:

```
$tabla = array ("Sauce", "Pino", "Naranja");  
$tabla[3]="Algarrobo";
```

En arrays asociativos:

```
$estadios_futbol = array("Valencia" => "Mestalla", "Real Sociedad" => "Anoeta");  
$estadios_futbol["Barcelona"] = "Nou Camp";
```

Veremos más adelante otras posibilidades del trabajo con arrays.

Ponemos a vuestra disposición las [páginas PHP que contienen los códigos con los que hemos trabajado](#).

**Referencia:** en el taller de PHP tenemos artículos sobre el trabajo con arrays en PHP. Concretamente tenemos uno que explica los distintos tipos de ordenación de arrays en PHP: [Ordenar arrays con PHP](#)

*Artículo por Miguel Angel Alvarez*

## 4.3.- Cadenas

*Aspectos relevantes de este tipo de variables. Lista de caracteres protegidos.*

Una de las variables más corrientes a las que tendremos que hacer frente en la mayoría de nuestros scripts son las cadenas, que no son más que información de carácter no numérico (textos, por ejemplo).

Para asignar a una variable un contenido de este tipo, lo escribiremos entre comillas dando lugar a declaraciones de este tipo:

```
$cadena="Esta es la información de mi variable"
```

Si queremos ver en pantalla el valor de una variable o bien un mensaje cualquiera usaremos la instrucción *echo* como ya lo hemos visto anteriormente:

```
echo $cadena //sacaría "Esta es la información de mi variable"
```

```
echo "Esta es la información de mi variable" //daría el mismo resultado
```

Podemos yuxtaponer o concatenar varias cadenas poniendo para ello un punto entre ellas:

```
<?
$cadena1="Perro";
$cadena2=" muerde";
$cadena3=$cadena1.$cadena2;
echo $cadena3 //El resultado es: "Perro muerde"
?>
```

### Ejecutar script

También podemos introducir variables dentro de nuestra cadena lo cual nos puede ayudar mucho en el desarrollo de nuestros scripts. Lo que veremos no es el nombre, sino el valor de la variable:

```
<?
$a=55;
$mensaje="Tengo $a años";
echo $mensaje //El resultado es: "Tengo 55 años"
?>
```

### Ejecutar script

La pregunta que nos podemos plantear ahora es...¿Cómo hago entonces para que en vez del valor "55" me salga el texto "\$a"? En otras palabras, cómo se hace para que el símbolo \$ no defina una variable sino que sea tomado tal cual. Esta pregunta es tanto más interesante cuanto que en algunos de scripts este símbolo debe ser utilizado por una simple razón comercial (pago en dólares por ejemplo) y si lo escribimos tal cual, el ordenador va a pensar que lo que viene detrás es una variable siendo que no lo es.

Pues bien, para meter éste y otros caracteres utilizados por el lenguaje dentro de las cadenas y no confundirlos, lo que hay que hacer es escribir una contrabarra delante:

Carácter	Efecto en la cadena
\$	Escribe dólar en la cadena
"	Escribe comillas en la cadena
\	Escribe contrabarra en la cadena
8/2	Escribe 8/2 y no 4 en la cadena

Además, existen otras utilidades de esta contrabarra que nos permiten introducir en nuestro documento HTML determinados eventos:

Carácter	Efecto en la cadena
t	Introduce una tabulación en nuestro texto
n	Cambiamos de línea

r

Retorno de carro

Estos cambios de línea y tabulaciones tienen únicamente efecto en el código y no en el texto ejecutado por el navegador. En otras palabras, si queremos que nuestro texto ejecutado cambie de línea hemos de introducir un *echo* "<br>" y no *echo* "\n" ya que este último sólo cambia de línea en el archivo HTML creado y enviado al navegador cuando la página es ejecutada en el servidor. La diferencia entre estos dos elementos puede ser fácilmente comprendida mirando el código fuente producido al ejecutar este script:

```
<HTML>
<HEAD>
<TITLE>cambiolinea.php</TITLE>
</HEAD>
<BODY>
<?
echo "Hola, n sigo en la misma línea ejecutada pero no en código fuente.<br>Ahora cambio de línea
ejecutada pero continuo en la misma en el código fuente."
?>
</BODY>
</HTML>
```

### Ejecutar script

Echar un vistazo al código fuente del navegador

El código fuente que observaríamos sería el siguiente:

```
<HTML>
<HEAD>
<TITLE>cambiolinea.php</TITLE>
</HEAD>
<BODY>
Hola,
    sigo en la misma línea ejecutada pero no en código fuente.<br>Ahora cambio de línea ejecutada pero
continuo en la misma en el código fuente.</BODY>
</HTML>
```

Las cadenas pueden asimismo ser tratadas por medio de funciones de todo tipo. Veremos estas funciones más adelante con más detalle. Tan sólo debemos retener que existen muchas posibles acciones que podemos realizar sobre ellas: Dividir las palabras, eliminar espacios sobrantes, localizar secuencias, reemplazar caracteres especiales por su correspondiente en HTML o incluso extraer las etiquetas META de una página web.

Artículo por *Rubén Álvarez*

## Parte 5:

# Funciones en PHP

Las funciones son esenciales para poder realizar código de calidad, tanto en PHP como en muchos otros lenguajes de programación. En estos capítulos del Manual de PHP aprenderemos a definir funciones, trabajar con parámetros y retornar valores.

### 5.1.- Funciones en PHP

*Utilidad de las funciones, creación y almacenamiento en archivos. Ejemplo práctico de creación de función.*

En nuestro manual de páginas dinámicas vimos el [concepto de función](#). Vimos que la función podría ser definida como un conjunto de instrucciones que explotan ciertas variables para realizar una tarea más o menos elemental.

PHP basa su eficacia principalmente en este tipo de elemento. Una gran librería que crece constantemente, a medida que nuevas versiones van surgiendo, es complementada con las funciones de propia cosecha dando como resultado un sinnúmero de recursos que son aplicados por una simple llamada.

Las funciones integradas en PHP son muy fáciles de utilizar. Tan sólo hemos de realizar la llamada de la forma apropiada y especificar los parámetros y/o variables necesarios para que la función realice su tarea.

Lo que puede parecer ligeramente más complicado, pero que resulta sin lugar a dudas muy práctico, es crear nuestras propias funciones. De una forma general, podríamos crear nuestras propias funciones para conectarnos a una base de datos o crear los encabezados o etiquetas meta de un documento HTML. Para una aplicación de comercio electrónico podríamos crear por ejemplo funciones de cambio de una moneda a otra o de cálculo de los impuestos a añadir al precio de artículo. En definitiva, es interesante crear funciones para la mayoría de acciones más o menos sistemáticas que realizamos en nuestros programas.

Aquí daremos el ejemplo de creación de una función que, llamada al comienzo de nuestro script, nos crea el encabezado de nuestro documento HTML y coloca el título que queremos a la página:

```
<?
function hacer_encabezado($titulo)
{
    $encabezado="<html><head>t<title>$titulo</title></head>";
    echo $encabezado;
}
?>
```

Esta función podría ser llamada al principio de todas nuestras páginas de la siguiente forma:

```
$titulo="Mi web";
hacer_encabezado($titulo);
```

De esta forma automatizamos el proceso de creación de nuestro documento. Podríamos por ejemplo incluir en la función otras variables que nos ayudasen a construir la etiquetas meta y de esta forma, con un esfuerzo mínimo, crearíamos los encabezados personalizados para cada una de nuestras páginas. De este mismo modo nos es posible crear cierres de documento o formatos diversos para nuestros textos como si se tratase de hojas de estilo que tendrían la ventaja de ser reconocidas por todos los navegadores.

Por supuesto, la función ha de ser definida dentro del script ya que no se encuentra integrada en PHP sino que la hemos creado nosotros. Esto en realidad no pone ninguna pega ya que puede ser incluida desde un archivo en el que iremos



almacenando las definiciones de las funciones que vayamos creando o recopilando.

Estos archivos en los que se guardan las funciones se llaman librerías. La forma de incluirlos en nuestro script es a partir de la instrucción *require* o *include*:

```
require("libreria.php") o include("libreria.php")
```

En resumen, la cosa quedaría así:

Tendríamos un archivo `libreria.php` como sigue

```
<?
//función de encabezado y colocación del título
function hacer_encabezado($titulo)
{
$encabezado="<html>n<head>nt<title>$titulo</title>n</head>n";
echo $encabezado;
}
?>
```

Por otra parte tendríamos nuestro script principal `página.php` (por ejemplo):

```
<?
include("libreria.php");
$titulo="Mi Web";
hacer_encabezado($titulo);
?>
<body>
El cuerpo de la página
</body>
</html>
```

### Ejecutar script

Echar un vistazo al código fuente del navegador

Podemos meter todas las funciones que vayamos encontrando dentro de un mismo archivo pero resulta muchísimo más ventajoso ir clasificándolas en distintos archivos por temática: Funciones de conexión a bases de datos, funciones comerciales, funciones generales, etc. Esto nos ayudara a poder localizarlas antes para corregirlas o modificarlas, nos permite también cargar únicamente el tipo de función que necesitamos para el script sin recargar éste en exceso además de permitirnos utilizar un determinado tipo de librería para varios sitios webs distintos.

También puede resultar muy práctico el utilizar una nomenclatura sistemática a la hora de nombrarlas: Las funciones comerciales podrían ser llamadas `com_loquesea`, las de bases de datos `bd_loquesea`, las de tratamiento de archivos `file_loquesea`. Esto nos permitirá reconocerlas enseguida cuando leamos el script sin tener que recurrir a nuestra oxidada memoria para descubrir su utilidad.

No obstante, antes de lanzarnos a crear nuestra propia función, merece la pena echar un vistazo a la [documentación](#) para ver si dicha función ya existe o podemos aprovecharnos de alguna de las existentes para aligerar nuestro trabajo. Así, por ejemplo, existe una función llamada `header` que crea un encabezado HTML configurable lo cual nos evita tener que crearla nosotros mismos.

Como puede verse, la tarea del programador puede en algunos casos parecerse a la de un coleccionista. Hay que ser paciente y metódico y al final, a base de trabajo propio, intercambio y tiempo podemos llegar a poseer nuestro pequeño tesoro.

**Nota:** Si lo deseas puedes repasar todos los conceptos anteriores sobre las funciones, así como diversas otras cosas interesantes en el [Videotutorial sobre las funciones en PHP](#).

## 5.1.1.- Ejemplo de función

Vamos a ver un ejemplo de creación de funciones en PHP. Se trata de hacer una función que recibe un texto y lo escribe en la página con cada carácter separado por "-". Es decir, si recibe "hola" debe escribir "h-o-l-a" en la página web.

**Nota:** Para comprender este ejemplo necesitamos conocer el bucle `for`, que se explica en el capítulo [Control del flujo en PHP: Bucles II](#).

La manera de realizar esta función será recorrer el string, caracter a caracter, para imprimir cada uno de los caracteres, seguido de el signo "-". Recorreremos el string con un bucle for, desde el carater 0 hasta el número de caracteres total de la cadena.

El número de caracteres de una cadena se obtiene con la función predefinida en PHP `strlen()`, que recibe el string entre paréntesis y devuelve el número de los caracteres que tenga.

```
<html>
<head>
    <title>funcion 1</title>
</head>

<body>

<?
function escribe_separa($cadena){
    for ($i=0;$i<strlen($cadena);$i++){
        echo $cadena[$i];
        if ($i<strlen($cadena)-1)
            echo "-";
    }
}

escribe_separa ("hola");
echo "<p>";
escribe_separa ("Texto más largo, a ver lo que hace");
?>
</body>
</html>
```

La función que hemos creado se llama `escribe_separa` y recibe como parámetro la cadena que hay que escribir con el separador "-". El bucle for nos sirve para recorrer la cadena, desde el primer al último carácter. Luego, dentro del bucle, se imprime cada carácter separado del signo "-". El if que hay dentro del bucle for comprueba que el actual no sea el último carácter, porque en ese caso no habría que escribir el signo "-" (queremos conseguir "h-o-l-a" y si no estuviera el if obtendríamos "h-o-l-a-").

En el código mostrado se hacen un par de llamadas a la función para ver el resultado obtenido con diferentes cadenas como parámetro. Podemos [ver el script en marcha](#).

*Artículo por Rubén Álvarez*

## 5.2.- Más sobre funciones: paso de parámetros

*Este capítulo pretende ser una ampliación de detalles que atañen al artículo dedicado a las funciones en PHP.*

Vamos a explicar algunos detalles adicionales sobre la definición y uso de funciones, para ampliar el artículo de [funciones en php](#).

### Paso de parámetros

Los parámetros son los datos que reciben las funciones y que utilizan para realizar las operaciones de la función. Una función puede recibir cualquier número de parámetros, incluso ninguno. A la hora de definir la función, en la cabecera, se definen los parámetros que va a recibir.

```
function f1 ($parametro1, $parámetro2)
```

Así definimos una función llamada `f1` que recibe dos parámetros. Como se puede observar, no se tiene que definir el tipo de datos de cada parámetro.

Los parámetros tienen validez durante la ejecución de la función, es decir, tienen un ámbito local a la función donde se están recibiendo. Cuando la función se termina, los parámetros dejan de existir.

### Los parámetros se pasan por valor

El paso de parámetros en PHP se realiza por valor. "Por valor" es una manera típica de pasar parámetros en funciones, quiere decir que el cambio de un dato de un parámetro no actualiza el dato de la variable que se pasó a la función. Por ejemplo, cuando invocamos una función pasando una variable como parámetro, a pesar de que cambiemos el valor del parámetro dentro de la función, la variable original no se ve afectada por ese cambio. Puede que se vea mejor con un ejemplo:

```
function porvalor ($parametro1){
    $parametro1="hola";
    echo "<br>" . $parametro1; //imprime "hola"
}

$mivARIABLE = "esto no cambia";
porvalor ($mivARIABLE);
echo "<br>" . $mivARIABLE; //imprime "esto no cambia"
```

Esta página tendrá como resultado:

hola  
esto no cambia

### Paso de parámetros por referencia

En contraposición al paso de parámetros por valor, está el paso de parámetros por referencia. En este último caso, el cambio del valor de un parámetro dentro de una función sí afecta al valor de la variable original.

Podemos pasar los parámetros por referencia si, en la declaración de la función, colocamos un "&" antes del parámetro.

```
<?
function porreferencia(&$cadena)
{
    $cadena = 'Si cambia';
}
$str = 'Esto es una cadena';
porreferencia ($str);
echo $str; // Imprime 'Si cambia'
?>
```

Este script mostrará por pantalla 'Si cambia'.

### Parámetros por defecto

Podemos definir valores por defecto para los parámetros. Los valores por defecto sirven para que los parámetros contengan un dato predefinido, con el que se inicializarán si no se le pasa ningún valor en la llamada de la función. Los valores por defecto se definen asignando un dato al parámetro al declararlo en la función.

```
function pordefecto ($parametro1="pepe";$parametro2=3)
```

Para la definición de función anterior, \$parametro1 tiene como valor por defecto "pepe", mientras que \$parametro2 tiene 3 como valor por defecto.

Si llamamos a la función sin indicar valores a los parámetros, estos tomarán los valores asignados por defecto:

```
pordefecto () // $parametro1 vale "pepe" y $parametro2 vale 3
```

Si llamamos a la función indicando un valor, este será tenido en cuenta para el primer parámetro.

```
pordefecto ("hola") // $parametro1 vale "hola" y $parametro2 vale 3
```

Atención, estamos obligados a declarar todos los parámetros con valores por defecto al final.

Artículo por *Miguel Ángel Álvarez*

## 5.3.- Más sobre funciones: Retorno de valores

*Para saber las formas en que una función devuelve valores este artículo contiene varias formas aptas para hacerlo.*

Las funciones pueden retornar valores. Para ello se utiliza la palabra "return" indicando a continuación el dato o variable que tienen que retornar. La función puede tener múltiples return, aunque sólo devolverá datos por uno de ellos cada vez porque, cuando se llama a return, se termina la ejecución de la función devolviendo el dato indicado.

### Ejemplo de función IVA

Vamos a ver un nuevo ejemplo para ilustrar el funcionamiento de una función un poco más avanzada, que utiliza parte de los nuevos conceptos introducidos en este artículo.

Se trata de hacer una función que calcula el IVA y que recibe dos parámetros. Uno el valor sobre el que se calcula y el otro el porcentaje a aplicar. Si no se indica el porcentaje de IVA se entiende que es el 16%.

```
<html>
<head>
  <title>ejemplo IVA</title>
</head>

<body>
<?
function iva($base,$porcentaje=16){
    return $base * $porcentaje /100;
}

echo iva(1000) . "<br>";
echo iva(1000,7) . "<br>";
echo iva(10,0) . "<br>";
?>

</body>
</html>
```

Si se han entendido bien los conceptos, este ejemplo no puede resultar difícil. La función recibe un parámetro llamado \$porcentaje con 16 como valor por defecto. Devuelve el porcentaje dado aplicado a la base también indicada por parámetro.

Así pues, en la primera ejecución de la función, como no se indica el porcentaje, se mostrará el 16% de 1000. En la segunda, se muestra el 7% de mil y en la tercera, el 0% de 10.

Puede verse el resultado en una página aparte. [http://www.desarrolloweb.com/articulos/ejemplos/php/ej\\_iva.php](http://www.desarrolloweb.com/articulos/ejemplos/php/ej_iva.php)

### Retornar múltiples valores

Una función devuelve un único valor. Si queremos hacer que se puedan devolver varios valores distintos tenemos que recurrir a un truco que consiste en devolver un array.

```
function small_numbers()
{
return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

list() se usa para asignar una lista de variables en una sola operación. Después de esa operación, \$zero valdrá 0, \$one valdrá 1 y \$two valdrá 2.

Artículo por *Miguel Angel Alvarez*

## Parte 6:

# Estructuras de control en PHP

Vemos una a una las distintas estructuras de control del flujo de los programas disponibles en el lenguaje de programación PHP: condicionales y bucles.

### 6.1.- Control del flujo en PHP: Condiciones IF

*Presentamos una de las herramientas principales usadas para controlar el flujo de nuestros scripts: Los condicionales IF.*

La programación exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución.

Como ejemplo, podríamos hacer alusión a un script que ejecute una secuencia diferente en función del día de la semana en el que nos encontramos.

Este tipo de acciones pueden ser llevadas a cabo gracias a una paleta de instrucciones presentes en la mayoría de los lenguajes. En este capítulo describiremos someramente algunas de ellas propuestas por PHP y que resultan de evidente utilidad.

Para evitar el complicar el texto, nos limitaremos a introducir las más importantes dejando de lado otras cuantas que podrán ser fácilmente asimilables a partir de ejemplos prácticos.

#### 6.1.1.- Las condiciones if

Cuando queremos que el programa, llegado a un cierto punto, tome un camino concreto en determinados casos y otro diferente si las condiciones de ejecución difieren, nos servimos del conjunto de instrucciones *if*, *else* y *elseif*. La estructura de base de este tipo de instrucciones es la siguiente:

```
if (condición)
{
    Instrucción 1;
    Instrucción 2;
    ...
}
else
{
    Instrucción A;
    Instrucción B;
    ...
}
```

Llegados a este punto, el programa verificará el cumplimiento o no de la condición. Si la condición es cierta las instrucciones 1 y 2 serán ejecutadas. De lo contrario (*else*), las instrucciones A y B serán llevadas a cabo.

Esta estructura de base puede complicarse un poco más si tenemos cuenta que no necesariamente todo es blanco o negro y que muchas posibilidades pueden darse. Es por ello que otras condiciones pueden plantearse dentro de la condición principal. Hablamos por lo tanto de condiciones anidadas que tendrían una estructura del siguiente tipo:

```
if (condición1)
{
    Instrucción 1;
    Instrucción 2;
    ...
}
else
{
    if (condición2)
    {
        Instrucción A;
        Instrucción B;
        ...
    }
    else
    {
        Instrucción X
        ...
    }
}
```

De este modo podríamos introducir tantas condiciones como queramos dentro de una condición principal.

De gran ayuda es la instrucción *elseif* que permite en una sola línea introducir una condición adicional. Este tipo de instrucción simplifica ligeramente la sintaxis que acabamos de ver:

```
if (condición1)
{
    Instrucción 1;
    Instrucción 2;
    ...
}
elseif (condición2)
{
    Instrucción A;
    Instrucción B;
    ...
}
else
{
    Instrucción X
    ...
}
```

El uso de esta herramienta resultará claro con un poco de práctica. Pongamos un ejemplo sencillo de utilización de condiciones. El siguiente programa permitiría detectar la lengua empleada por el navegador y visualizar un mensaje en dicha lengua.

```
<HTML>
<HEAD>
```

```
<TITLE>Detector de Lengua</TITLE>
</HEAD>
<BODY>
<?
//Antes de nada introducimos mensajes en forma de variables
$espanol="Hola";
$ingles="Hello";
$frances="Bonjour";

//Ahora leemos del navegador cuál es su lengua oficial
$id idioma=substr($_SERVER['HTTP_ACCEPT_LANGUAGE'],0,2);

//Formulamos las posibilidades que se pueden dar
if ($idioma == "es")
{echo "$espanol";}
elseif ($idioma=="fr")
{echo "$frances";}
else
{echo "$ingles";}
?>
</BODY>
</HTML>
```

### Ejecutar script

Para poder ver el funcionamiento de este script es necesario cambiar el idioma preferido lo cual puede ser realizado a partir del menú de opciones del navegador.

Para leer la lengua aceptada por el navegador lo que hacemos es definir una variable (*\$idioma*) y, mediante la función *substr*, recogemos las dos primeras letras del código correspondiente al idioma aceptado por el navegador (*\$\_SERVER['HTTP\_ACCEPT\_LANGUAGE']*).

La tercera parte de script se encarga de ver si el navegador está en español (es), francés (fr) o en cualquier otro idioma que no sea ninguno de estos dos y de imprimir el mensaje que proceda en cada caso.

A notar que, cuando se trata de comparar variables, ponemos un doble igual "==" en lugar de un simple "=". Este último queda reservado exclusivamente para asignar valores a variables

**Referencia:** Hemos publicado un vídeo para mostrar la creación y el funcionamiento de las estructuras IF: [Estructuras de control, Vídeo 1: condicional if](#)

Artículo por *Rubén Álvarez*

## 6.2.- Control del flujo en PHP: Bucles I

*Estructura y funcionamiento de los bucles while y do/while.*

Los ordenadores, como cualquier máquina, están diseñados para realizar tareas repetitivas. Es por ello que nuestros programas pueden aprovecharse de este principio para realizar una determinada secuencia de instrucciones un cierto número de veces. Para ello, utilizamos las estructuras llamadas en bucle que nos ayudan a, usando unas pocas líneas, realizar una tarea incluida dentro del bucle un cierto número de veces definido por nosotros mismos.

PHP propone varios tipos de bucle cada uno con características específicas:

### 6.2.1.- Bucle while

Sin duda el bucle más utilizado y el más sencillo. Lo usamos para ejecutar las instrucciones contenidas en su interior siempre y cuando la condición definida sea verdadera. La estructura sintáctica es la siguiente.

```
while (condición)
```

```
{  
    instruccion1;  
    instruccion2;  
    ...  
}
```

Un ejemplo sencillo es este bucle que aumenta el tamaño de la fuente en una unidad a cada nueva vuelta por el bucle:

```
<?  
$size=1;  
While ($size<=6)  
{  
    echo"<font size=$size>Tamaño $size</font><br>n";  
    $size++;  
}  
?>
```

### Ejecutar script

A modo de explicación, diremos que, antes de nada, hemos de definir el valor de la variable que vamos a evaluar en la condición. Algo absolutamente obvio pero fácil de olvidar. En este caso le hemos atribuido un valor de 1 que corresponde a la letra más pequeña.

El paso siguiente es crear el bucle en el que imponemos la condición que la variable no exceda el valor de 6.

La instrucción a ejecutar será imprimir en nuestro documento un código HTML en el que la etiqueta *font* y el mensaje que contiene varían a medida que *\$size* cambia su valor.

El siguiente paso es incrementar en una unidad el valor de *\$size*. Esto se puede hacer con una expresión como la mostrada en el bucle (*\$size++*) que en realidad es sinónima de:

```
$size=$size+1
```

Veremos otras de estas abreviaciones más adelante.

## 6.2.2.- Otro ejemplo del bucle While

El bucle while se suele utilizar cuando no se sabe exactamente cuantas iteraciones se deben realizar antes de acabar. Vamos a utilizarlo en otro ejemplo, en el que hay que recorrer una cadena hasta encontrar un carácter dado. Si lo encuentra, escribir su posición. Si no, escribir que no se ha encontrado.

**Nota:** Para hacer este ejercicio necesitamos conocer la función de cadena `strlen()`, que obtiene la longitud de la cadena que se le pase por parámetro.

`int strlen (string cad)`  
Devuelve un entero igual a la longitud de la cadena.

```
<?  
$cadena = "hola a todo el mundo";  
  
//recorro la cadena hasta encontrar una "m"  
$i=0;  
while ($cadena[$i]!="m" && $i< strlen($cadena)){  
    $i++;  
}  
  
if ($i==strlen($cadena))  
    echo "No se encuentra...";  
else  
    echo "Está en la posición $i";  
?>
```

En este ejemplo se define una cadena con el valor "hola a todo el mundo". Posteriormente se recorre esa cadena hasta el final de la cadena o hasta encontrar el carácter "m", utilizando una variable *\$i* que lleva la cuenta de los caracteres recorridos.

Al final del bucle while, si se salió porque se encontró el carácter "m", la variable *\$i* valdrá un número menor que la longitud



de la cadena. Si se salió por llegar al final de la cadena, la variable \$i valdrá lo mismo que la longitud en caracteres de esa cadena. En el condicional simplemente se comprueba si \$i vale o no lo mismo que la longitud de la cadena, mostrando los mensajes adecuados en cada caso.

Podemos [ver el ejemplo en funcionamiento](#).

### 6.2.3.- Bucle do/while

Este tipo de bucle no difiere en exceso del anterior. La sintaxis es la siguiente:

```
do
{
    instruccion1;
    instruccion2;
    ...
}
while (condición)
```

La diferencia con respecto a los bucles *while* es que este tipo de bucle evalúa la condición al final con lo que, incluso siendo falsa desde el principio, éste se ejecuta al menos una vez.

**Referencia:** Para una ayuda práctica sobre estos bucles ver el siguiente videotutorial [Estructuras de control, Vídeo 2: bucles for, while y do-while](#). Además, las explicaciones sobre estructuras de control se complementan con el [videotutorial de Switch y las instrucciones break y continue](#).

Artículo por *Rubén Álvarez*

## 6.3.- Control del flujo en PHP: Bucles II

*Estructura y funcionamiento de los bucles for y foreach. Explicamos como salir de un bucle: Break y continue*

Este es el segundo artículo sobre los bucles en PHP que publicamos en el [Manual de PHP](#). El anterior artículo explicó acerca de los [bucles while y do-while](#). Así pues, ahora vamos a dedicarnos a los otros tipos de bucles, que son el bucle for, para iterar un número dado de veces, foreach, útil para recorrer arrays, así como las sentencias break y continue.

### 6.3.1.- Bucle for

PHP está provisto de otros tipos de bucle que también resultan muy prácticos en determinadas situaciones. El más popular de ellos es el bucle *for* que, como para los casos anteriores, se encarga de ejecutar las instrucciones entre llaves. La diferencia con los anteriores radica en cómo se plantea la condición de finalización del bucle. Para aclarar su funcionamiento vamos a expresar el ejemplo de bucle *while* visto en el capítulo anterior en forma de bucle *for*:

```
<?
For ($size=1;$size<=6;$size++)
{
    echo"<font size=$size>Tamaño $size</font><br>n";
}
?>
```

#### [Ejecutar script](#)

Las expresiones dentro del paréntesis definen respectivamente:

- Inicialización de la variable. Valida para la primera vuelta del bucle.
- Condición de evaluación a cada vuelta. Si es cierta, el bucle continua.

-Acción a realizar al final de cada vuelta de bucle.

### 6.3.2.- Bucle foreach

Este bucle, implementado en las versiones de PHP4, nos ayuda a recorrer los valores de un array lo cual puede resultar muy útil por ejemplo para efectuar una lectura rápida del mismo. Recordamos que un array es una variable que guarda un conjunto de elementos (valores) catalogados por claves.

La estructura general es la siguiente:

```
Foreach ($array as $clave=>$valor)
{
    instruccion1;
    instruccion2;
    ...;
}
```

Un ejemplo práctico es la lectura de un array lo cual podría hacerse del siguiente modo:

```
<?
$moneda=array("España"=> "Peseta","Francia" => "Franco","USA" => "Dolar");
Foreach ($moneda as $clave=>$valor)
{
    echo "Pais: $clave Moneda: $valor<br>";
}
?>
```

#### Ejecutar script

Este script se encargaría de mostrarnos por pantalla el contenido del array *\$moneda*. No resultaría mala idea crear una función propia basada en este bucle que nos permitiese visualizar arrays monodimensionales y almacenarla en nuestra librería. Esta función podría ser definida de esta forma:

```
Function mostrar_array ($array)
{
    foreach ($array as $clave=>$valor)
    {echo "$clave=>$valor<br>";}
}
```

### 6.3.3.- Break y continue

Estas dos instrucciones se introducen dentro de la estructura y nos sirven respectivamente para escapar del bucle o saltar a la iteración siguiente. Pueden resultarnos muy prácticas en algunas situaciones.

Durante una iteración de un bucle podemos saltar directamente a la siguiente iteración, sin seguir con la actual, con la instrucción *continue*.

También podemos detener completamente las repeticiones de cualquier bucle con *break*, lo que parará la ejecución de la iteración actual y de las siguientes que pudiera haber.

**Referencia:** Si deseas ver la construcción de un bucle *for* y otros tipos de bucles, en vídeo te recomendamos el siguiente videotutorial [Estructuras de control, Vídeo 2: bucles for, while y do-while](#).

Además, si deseas aprender la estructura de control *switch* y las instrucciones *break* y *continue*, así como ver otro ejemplo de *foreach*, te recomendamos ver los [Videotutoriales de estructuras de control en PHP II](#).

Artículo por *Rubén Álvarez*

## Parte 7:

# Aplicaciones web: paso de variables y memoria de estados

En los siguientes capítulos veremos diversas maneras que existen en PHP para conseguir que un conjunto de páginas y scripts se comporten como una aplicación web. Veremos cómo pasar datos de unas páginas a otras, por POST y GET, cómo memorizar datos asociados a un usuario a lo largo de toda la sesión y cómo memorizar datos en cookies, que perdurarán entre varias sesiones.

### 7.1.- Pasar variables por la URL con PHP

*Veremos cómo transferir variables de una página a otra por medio de la URL, en PHP.*

Bucles y condiciones son muy útiles para procesar los datos dentro de un mismo script. Sin embargo, en un sitio Internet, las páginas vistas y los scripts utilizados son numerosos. Muy a menudo necesitamos que nuestros distintos scripts estén conectados unos con otros y que se sirvan de variables comunes. Por otro lado, el usuario interacciona por medio de formularios cuyos campos han de ser procesados para poder dar una respuesta. Todo este tipo de factores dinámicos han de ser eficazmente regulados por un lenguaje como PHP.

Es posible que ya os hayáis percatado de que las variables de un script tienen una validez exclusiva para el script y que nos resulta imposible conservar su valor cuando ejecutamos otro archivo distinto aunque ambos estén enlazados. Existen varias formas de enviar las variables de una página a otra de manera a que la página destino reconozca el valor asignado por el script de origen:

#### 7.1.1.- Pasar variables por URL

Para pasar las variables de una página a otra lo podemos hacer introduciendo dicha variable dentro del enlace hipertexto de la página destino. La sintaxis sería la siguiente:

```
<a href="destino.php?variable1=valor1&variable2=valor2&...">Mi enlace</a>
```

Podéis observar que estas variables no poseen el símbolo \$ delante. Esto es debido a que en realidad este modo de pasar variables no es específico de PHP sino que es utilizado por otros lenguajes.

Ahora nuestra variable pertenece también al entorno de la página *destino.php* y está lista para su explotación.

**Nota:** No siempre se definen automáticamente las variables recibidas por parámetro en las páginas web, depende de una variable de configuración de PHP: `register_globals`, que tiene que estar activada para que así sea. Ver comentarios del artículo al final de la página para más información.

Para aclarar posibles dudas, veamos esto en forma de ejemplo. Tendremos pues dos páginas, *origen.html* (no es necesario darle extensión PHP puesto que no hay ningún tipo de código) y *destino.php*:

```
<HTML>
<HEAD>
<TITLE>origen.html</TITLE>
</HEAD>
<BODY>
<a href="destino.php?saludo=hola&texto=Esto es una variable texto">Paso variables saludo y texto a
la página destino.php</a>
</BODY>
</HTML>

<HTML>
<HEAD>
<TITLE>destino.php</TITLE>
</HEAD>
<BODY>
<?
echo "Variable $saludo: $saludo <br>n";
echo "Variable $texto: $texto <br>n"
?>
</BODY>
</HTML>
```

[Ejecutar ejemplo](#)

### 7.1.2.- \$HTTP\_GET\_VARS

Recordamos que es posible recopilar en una variable tipo array el conjunto de variables que han sido enviadas al script por este método a partir de la variable de sistema `$HTTP_GET_VARS`, que es un array asociativo. Utilizándolo quedaría así:

```
<?
echo "Variable $saludo: $HTTP_GET_VARS["saludo"] <br>n";
echo "Variable $texto: $HTTP_GET_VARS["texto"] <br>n"
?>
```

**Nota:** Aunque podamos recoger variables con este array asociativo o utilizar directamente las variables que se definen en nuestra página, resulta más seguro utilizar `$HTTP_GET_VARS` por dos razones, la primera que así nos aseguramos que esa variable viene realmente de la URL y la segunda, que así nuestro código será más claro cuando lo volvamos a leer, porque quedará especificado que esa variable estamos recibiendo por la URL.

### 7.1.3.- \$\_GET

A partir de la versión 4.1.0 de PHP se ha introducido el array asociativo `$_GET`, que es idéntico a `$HTTP_GET_VARS`, aunque un poco más corto de escribir.

### 7.1.4.- Caracteres especiales en URL y su codificación con PHP

Hay algunos caracteres raros que no se pueden pasar, tal cual, por la URL. Por ejemplo, una URL no puede contener espacios en blanco, por lo que si intentas enviar una variable por URL con un valor que tiene un espacio en blanco, te dará problemas. Por ejemplo, el signo "\*" no puede figurar tampoco en una URL. Así pues, tenemos que hacer algo para convertir esos caracteres, de modo que no den problemas en la URL.

La solución en PHP es sencilla, simplemente debemos codificar la variable que tiene caracteres conflictivos a formato URL. Para ello utilizamos la función `urlencode()`, que viene en la librería de funciones de PHP. Podemos encontrar más información sobre esto en la FAQ: [Problemas con variables pasadas en la URL en algunos caracteres](#).

**Referencia:** Si lo deseas, puedes complementar esta información con unos explicativos [videotutoriales sobre el paso de variables por GET en PHP](#).

Artículo por *Rubén Álvarez*

## 7.2.- Procesar variables de formularios. POST en PHP

*Veremos cómo transferir variables con PHP, de una página a otra por medio de formularios, lo que se conoce habitualmente por POST.*

Este tipo de transferencia es de gran utilidad ya que nos permite interaccionar directamente con el usuario.

El proceso es similar al explicado para las URLs. Primeramente, presentamos una primera página con el formulario clásico a rellenar y las variables son recogidas en una segunda página que las procesa:

**Nota:** No siempre se definen automáticamente las variables recibidas por el formulario en las páginas web, depende de una variable de configuración de PHP: `register_globals`, que tiene que estar activada para que así sea. Ver comentarios del artículo al final de la página para más información.

```
<HTML>
<HEAD>
<TITLE>formulario.html</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="destino2.php">
Nombre<br>
<INPUT TYPE="TEXT" NAME="nombre"><br>
Apellidos<br>
<INPUT TYPE="TEXT" NAME="apellidos"><br>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>

<HTML>
<HEAD>
<TITLE>destino2.php</TITLE>
</HEAD>
<BODY>
<?
echo "Variable $nombre: $nombre <br>n";
echo "Variable $apellidos: $apellidos <br>n"
?>
</BODY>
</HTML>
```

[Ejecutar ejemplo](#)

### 7.2.1.- \$HTTP\_POST\_VARS

Recordamos que es posible recopilar en una variable tipo array el conjunto de variables que han sido enviadas al script por este método a partir de la variable de sistema `$HTTP_POST_VARS`.

```
echo "Variable $nombre: " . $HTTP_POST_VARS["nombre"] . "<br>n";
```

**Nota:** Aunque podamos recoger variables con este array asociativo o utilizar directamente las variables que se definen en nuestra página, resulta más seguro utilizar `$HTTP_POST_VARS` por dos razones, la primera que así nos aseguramos que esa variable viene realmente de un formulario y la segunda, que así nuestro código será más claro cuando lo volvamos a leer, porque quedará especificado que esa variable estamos recibiendo por un formulario.

### 7.2.2.- \$\_POST

A partir de PHP 4.1.0 se pueden recoger las variables de formulario utilizando también el array asociativo \$\_POST, que es el mismo que \$HTTP\_POST\_VARS, pero más corto de escribir.

### 7.2.3.- Ejemplo de restricción de acceso por edad

Para continuar aportando ejemplos al uso de formularios vamos a realizar una página que muestra solicita la edad del visitante y, dependiendo de dicha edad, permita o no visualizar el contenido de la web. A los mayores de 18 años se les permite ver la página y a los menores no.

El ejemplo es muy sencillo y no valdría tal cual está para utilizarlo a modo de una verdadera restricción de acceso. Únicamente nos sirve para saber cómo obtener datos de un formulario y como tratarlos para realizar una u otra acción, dependiendo de su valor.

La página del formulario, que hemos llamado edad.php tendría esta forma:

```
<html>
<head>
  <title>Restringir por edad</title>
</head>

<body>

<form action="edad2.php" method="post">
Escribe tu edad: <input type="text" name="edad" size="2">
<input type="submit" value="Entrar">
</form>

</body>
</html>
```

Esta es una página sin ningún código PHP, simplemente tiene un formulario. Fijémonos en el action del formulario, que está dirigido hacia una página llamada edad2.php, que es la que recibirá el dato de la edad y mostrará un contenido u otro dependiendo de ese valor. Su código es el siguiente:

```
<html>
<head>
  <title>Restringir por edad</title>
</head>

<body>

<?
$edad = $_POST["edad"];
echo "Tu edad: $edad<p>";

if ($edad < 18) {
  echo "No puedes entrar";
}else{
  echo "Bienvenido";
}
?>
</body>
</html>
```

Esperamos que este otro código tampoco resulte extraño. Simplemente se recibe la edad, utilizando el array \$\_POST. Luego se muestra la edad y se ejecuta una expresión condicional en función de que la edad sea menor que 18. En caso positivo (edad menor que 18), se muestra un mensaje que informa de que no se deja acceder al página. En caso negativo (mayor o igual a 18) se muestra un mensaje de bienvenida.

Podemos [ver el ejemplo en funcionamiento](#).

Artículo por *Rubén Álvarez*

## 7.3.- Autollamada de páginas

*Páginas que se llaman a si mismas pasando datos por POST o GET: formularios reentrantes y variables pasadas por URL a la misma página.*

Al incluir un formulario en una página se debe indicar, a través del atributo action, el nombre del archivo PHP al que enviaremos los datos escritos en el formulario. De este modo, para un esquema de envío de datos por formulario, pueden participar dos páginas: una que contiene el formulario y otra que recibe los datos de dicho formulario.

Lo mismo ocurre cuando enviamos variables por una URL. Tenemos una página que contendrá el enlace y otra página que recibirá y tratará esos datos para mostrar unos resultados.

En el presente artículo vamos a ver cómo se puede enviar y recibir datos de un formulario con una única página. Asimismo, veremos como en la misma página podemos tener enlaces con paso de variables por URL y además, podemos recoger y tratar esos datos con la misma página. A este efecto podemos llamarle "autollamada de páginas", también se le suele llamar como "Formularios reentrantes" o términos similares. Es muy interesante conocer el modo de funcionamiento de estos scripts, porque serán muy habituales en nuestras páginas PHP y ayudan mucho a tener los códigos ordenados.

En ambos casos, para formularios o envío de datos por la URL, se debe seguir un esquema como este:

- Comprobar si recibo datos por URL o por formulario
- Si no recibo datos
  - Muestro el formulario o los enlaces que pasan variables.
- Si recibo datos
  - Entonces tengo que procesar el formulario o las variables de la URL

### Para un formulario

Veamos a continuación como sería el código de un formulario reentrante.

```
<html>
<head>
  <title>Me llamo a mi mismo...</title>
</head>

<body>
<?
if (!$_POST){
?>
  <form action="auto-llamada.php" method="post">
    Nombre: <input type="text" name="nombre" size="30">
    <br>
    Empresa: <input type="text" name="empresa" size="30">
    <br>
    Telefono: <input type="text" name="telefono" size=14 value="+34 " >
    <br>
    <input type="submit" value="Enviar">
  </form>
<?
}else{
  echo "<br>Su nombre: " . $_POST["nombre"];
  echo "<br>Su empresa: " . $_POST["empresa"];
  echo "<br>Su Teléfono: " . $_POST["telefono"];
}
?>
</body>
</html>
```

En el ejemplo, el primer paso es conocer si se están recibiendo o no datos por un formulario. Para ello se comprueba con un

enunciado if si existe o no una variable \$\_POST.

En concreto if (!\$\_POST) querría decir algo como "Si no existen datos venidos de un formulario". En caso de que no existan, muestro el formulario. En caso de que sí existan, recojo los datos y los imprimo en la página.

Se puede [ver el ejemplo en funcionamiento](#) en una página aparte.

### Para paso de variables por URL

La idea es la misma. Comprobar con un enunciado if si se reciben o no datos desde una URL. Veamos el código a continuación. Se trata de una página que muestra una serie de enlaces para ver las tablas de multiplicar de el 1 hasta el 10. Cada uno de los enlaces muestra una tabla de multiplicar. Pulsando el primer enlace podemos ver la tabla del 1, pulsando el segundo la tabla del 2, etc.

Recordemos que la página se llama a si misma. Para comprenderla más fácilmente será interesante [verla en funcionamiento](#).

```
<html>
<head>    <title>Tablas de multiplicar</title>
</head>

<body>
<?
if (!$_GET){
    for ($i=1;$i<=10;$i++){
        echo "<br><a href='ver_tabla.php?tabla=$i'>Ver la tabla del $i</a>\n";
    }
} else {
    $tabla=$_GET["tabla"];
?>
    <table align=center border=1 cellpadding="1">
<?
    for ($i=0;$i<=10;$i++){
        echo "<tr><td>$tabla X $i</td><td>=</td><td>" . $tabla * $i . "</td></tr>\n";
    }
?>
    </table>
<?
}
?>
</body>
</html>
```

Este código es un poco más complicado, porque hace un poco más de cosas que el anterior, pero para el asunto que nos ocupa que es la autollamada de páginas, todo sigue igual de simple.

Hay que fijarse en el if que comprueba si se reciben o no datos por URL: if (!\$\_GET), que querría decir algo como "Si no se reciben variables por la URL".

En caso positivo (no se reciben datos por URL) se muestran los enlaces para ver cada una de las tablas y en caso de que sí se reciban datos, se muestra la tabla de multiplicar del número que se está recibiendo en la URL.

Para hacer para mostrar los enlaces y las tablas de multiplicar se utilizan bucles for, que esperamos que no resulten desconocidos para el lector. Puede conocerse algo más sobre los bucles for en [Control del flujo en PHP: Bucles II](#).

*Artículo por Miguel Angel Alvarez*

## 7.4.- Utilización de las cookies en PHP

*Aprendemos sobre las cookies en PHP. Explicamos en qué consisten estas célebres galletas y describimos*



### su empleo y utilidad.

Sin duda este término resultara familiar para muchos. Algunos lo habrán leído u oído pero no saben de qué se trata. Otros sin embargo sabrán que las cookies son unas informaciones almacenadas por un sitio web en el disco duro del usuario. Esta información es almacenada en un archivo tipo texto que se guarda cuando el navegador accede al sitio web.

**Referencia:** Una explicación de las cookies más detallada se puede encontrar en el artículo [Qué son las cookies](#), publicado en DesarrolloWeb.com.

Es posible, por supuesto, ver estos archivos. Para abrirlos hay que ir al directorio C:\Windows\Cookies para los usuarios de IE 4+ o a C:\...\Netscape\Users\defaultuser para usuarios de Netscape. Como podréis comprobar, en la mayoría de los casos la información que se puede obtener es indescifrable.

La utilidad principal de las cookies es la de poder identificar al navegador una vez éste visita el sitio por segunda vez y así, en función del perfil del cliente dado en su primera visita, el sitio puede adaptarse dinámicamente a sus preferencias (lengua utilizada, colores de pantalla, formularios rellenados total o parcialmente, redirección a determinadas páginas...).

Para crear **cookies con PHP**, modificar o generar una nueva cookie lo podemos hacer a partir de la función SetCookie:

```
setcookie("nombre_de_la_cookie", valor, expiracion);
```

Pongamos un ejemplo sencillo. Imaginemos que queremos introducir en una variable cookie el nombre del visitante. El nombre ha podido ser previamente recogido por un formulario tal y como hemos visto:

```
setcookie("persona", $nombre, time() + 86400 * 365);
```

De este modo hemos creado una cookie php llamada persona que tiene como valor el contenido de la variable \$nombre y tendrá una duración de 1 año a partir de su creación (el tiempo time() actual en segundos sumado a un año en segundos).

Es importante que la creación de la cookie sea previa a la apertura del documento HTML. En otras palabras, las llamadas a la función setcookie() deben ser colocadas antes de la etiqueta HTML.

Por otra parte, es interesante señalar que el hecho de que definir una cookie ya existente implica el borrado de la antigua. Del mismo modo, el crear una primera cookie conlleva la generación automática del archivo texto.

Para utilizar el valor de la cookie en nuestros scripts tan sólo tendremos que llamar la variable que define la cookie. ¡Realmente sencillo!

Hay que tener cuidado sin embargo de no definir variables en nuestro script con el mismo nombre que las cookies puesto que PHP privilegiará el contenido de la variable local con respecto a la cookie y no dará un mensaje de error. Esto nos puede conducir a errores realmente difíciles de detectar.

Recordamos que es posible recopilar en una variable tipo array el conjunto de cookies almacenadas en el disco duro del internauta mediante la variable de servidor \$HTTP\_COOKIE\_VARS

Las cookies son una herramienta fantástica para personalizar nuestra página pero hay que ser cautos ya que, por una parte, no todos los navegadores las aceptan y por otra, se puede deliberadamente impedir al navegador la creación de cookies. Es por ello que resultan un complemento y no una fuente de variables infalible para nuestro sitio. Como has podido ver, las **Cookies son muy sencillas de utilizar en PHP**.

Puedes profundizar más en la creación de cookies en el siguiente artículo: [Cookies en PHP](#).

Artículo por *Rubén Álvarez*

## 7.5.- Cookies en PHP

*Explicaciones completas sobre cookies en PHP, con todos los parámetros de la función setcookie() y el array \$\_COOKIE*

En este artículo del [Manual de PHP](#) vamos a demostraros que las **cookies en PHP** son muy fáciles de utilizar. Ya empezamos a explicar algunas claves interesantes sobre este asunto en el artículo [utilización de cookies](#), anterior entrega del temario de DesarrolloWeb.com sobre la programación en PHP.

En el presente artículo vamos a mostrar otros detalles que conviene saber para trabajar con las cookies. Tanto los procesos de creación como los de lectura, pero sobre todo vamos a realizar un estudio completo sobre los diferentes parámetros que tenemos disponibles a la hora de llamar a la función `setcookie()`, que sirve para dar de alta las galletitas en el navegador del usuario que visita nuestra web.

### 7.5.1.- Crear cookies en PHP

En **PHP las cookies se controlan por medio de una función**, que nos sirve para generarlas y guardarlas en el navegador del usuario. Es la función `setcookie()`, que recibe varios parámetros, entre ellos, el nombre de la cookie, el valor y la caducidad. El único parámetro obligatorio es el primero, el nombre de la cookie, los demás son opcionales.

Veamos la lista entera de parámetros de `setcookie()` con sus explicaciones:

#### Nombre

Un string con el nombre que queremos darle a la cookie a guardar. Puede ser cualquiera que deseemos.

#### Valor

Una cadena de caracteres que es el valor que va a tener la cookie.

#### Caducidad

Es un timestamp con el valor de la fecha en la que **caducará la cookie**. Lo normal es utilizar la función `time()`, que genera el timestamp actual y sumarle el número de segundos que quedamos que dure la cookie. Por ejemplo, `time() + (60 * 60 * 24 * 365)` haría que la cookie durase un año en el sistema del usuario.

#### Ruta

El camino o ruta donde la cookie se podrá utilizar dentro del dominio. Por defecto, la cookie se podrá utilizar en el directorio donde se ha creado y sus subdirectorios. Si indicamos "/" la cookie tendrá validez dentro de todo el dominio.

#### Dominio

Es el subdominio donde se podrá acceder a la cookie. Las cookies sólo se pueden generar y utilizar para el dominio de la página donde está colocado el script, pero podemos hacerlo visible para todos los subdominios del dominio de la web por medio de ".midominio.com".

#### Seguro

Es un booleano que, si es true, indica que la cookie sólo puede ser transmitida por https (http seguro).

#### Sólo http

Esto es otro booleano que sirve para indicar que la cookie sólo puede ser accedida por medio de las cabeceras del http, lo que la haría inalcanzable para lenguajes de script en el cliente como JavaScript. Este parámetro fue añadido en PHP 5.2.0

**La función `setcookie()` de PHP genera y envía la cookie al navegador** y devuelve un booleano, si es true indica que se pudo incluir en el navegador del usuario y si es false indica que no ha podido colocarla en el sistema. Pero este valor no indica que luego el visitante la haya aceptado o no, puesto que el navegador puede haberlo configurado para no aceptar cookies y esto no lo puede detectar `setcookie()` directamente.

Por ejemplo, estas serían diferentes llamadas a `setcookie()`:

```
setcookie("migalleta", "mivalor");  
setcookie("cookie2", "mivalor2", time() + 3600);  
setcookie("otracookie", "valorfinal", time() + 3600, "/", ".midominio.com");
```

Pero atención en un asunto: Para enviar una cookie al navegador se debe hacer antes de haber enviado las cabeceras del http al cliente, es decir, antes de haber escrito cualquier texto en la página. Si no, PHP podrá lanzar un error de headers already sent (cabeceras ya enviadas).

## Recuperar cookies con PHP

Por otra parte, para recibir las cookies que el navegador del usuario pueda tener creadas en el sistema se utiliza el array asociativo `$_COOKIE`. En este array están todas las cookies que tiene disponible la página PHP en el dominio y el directorio donde está colocado.

Por medio del nombre de la cookie accedemos a su valor:

```
$_COOKIE["migalleta"];  
$_COOKIE["cookie2"];
```

Para ver un ejemplo de uso de **cookies PHP** acceder al taller de PHP [Estilos CSS distintos a una página con PHP y cookies](#).

*Artículo por Miguel Angel Alvarez*

## 7.6.- Sesiones I

*Nos introducimos al concepto de sesión y aprendemos su manejo y funcionamiento.*

En los programas que hemos visto hasta ahora, hemos utilizado variables que sólo existían en el archivo que era ejecutado. Cuando cargábamos otra página distinta, los valores de estas variables se perdían a menos que nos tomásemos la molestia de pasarlos por la URL o inscribirlos en las cookies o en un formulario para su posterior explotación. Estos métodos, aunque útiles, no son todo lo prácticos que podrían en determinados casos en los que la variable que queremos conservar ha de ser utilizada en varios scripts diferentes y distantes los unos de los otros.

Podríamos pensar que ese problema puede quedar resuelto con las cookies ya que se trata de variables que pueden ser invocadas en cualquier momento. El problema, ya lo hemos dicho, es que las cookies no son aceptadas ni por la totalidad de los usuarios ni por la totalidad de los navegadores lo cual implica que una aplicación que se sirviera de las cookies para pasar variables de un archivo a otro no sería 100% infalible. Es importante a veces pensar en "la inmensa minoría", sobre todo en aplicaciones de comercio electrónico donde debemos captar la mayor cantidad de clientes posibles y nuestros scripts deben estar preparados ante cualquier eventual deficiencia del navegador del cliente.

Nos resulta pues necesario el poder declarar ciertas variables que puedan ser reutilizadas tantas veces como queramos dentro de una misma sesión. Imaginemos un sitio multilingüe en el que cada vez que queremos imprimir un mensaje en cualquier página necesitamos saber en qué idioma debe hacerse. Podríamos introducir un script identificador de la lengua del navegador en cada uno de los archivos o bien declarar una variable que fuese válida para toda la sesión y que tuviese como valor el idioma reconocido en un primer momento.

Pensemos también en un carrito de la compra de una tienda virtual donde el cliente va navegando por las páginas del sitio y añadiendo los artículos que quiere comprar a un carrito. Este carrito podría ser perfectamente una variable de tipo array (tabla) que almacena para cada referencia la cantidad de artículos contenidos en el carrito. Esta variable debería ser obviamente conservada continuamente a lo largo de todos los scripts.

Este tipo de situaciones son solventadas a partir de las variables de sesión. Una sesión es considerada como el intervalo de tiempo empleado por un usuario en recorrer nuestras páginas hasta que abandona nuestro sitio o deja de actuar sobre él durante un tiempo prolongado o bien, sencillamente, cierra el navegador.

PHP nos permite almacenar variables llamadas de sesión que, una vez definidas, podrán ser utilizadas durante este lapso de tiempo por cualquiera de los scripts de nuestro sitio. Estas variables serán específicas del usuario de modo que varias variables sesión del mismo tipo con distintos valores pueden estar coexistiendo para cada una de las sesiones que están teniendo lugar simultáneamente. Estas sesiones tienen además su propio identificador de sesión que será único y específico.

Algunas mejoras referentes al empleo de sesiones han sido introducidas con PHP4. Es a esta nueva versión a la que haremos referencia a la hora de explicar las funciones disponibles y la forma de operar. Para los programadores de PHP3 la diferencia mayor es que están obligados a gestionar ellos mismos las sesiones definir sus propios identificadores de sesión.

Veamos en el siguiente capítulo la forma de plasmar esta necesidad técnica en nuestros scripts a partir de las funciones que gestionan las sesiones en PHP.

Artículo por *Rubén Álvarez*

## 7.7.- Sesiones en PHP II

*Describimos los metodos de trabajo con sesiones en PHP, inicializar sesión, crear variables de sesión o recuperarlas.*

Las sesiones, en aplicaciones web realizadas con PHP y en el desarrollo de páginas web en general, nos sirven para almacenar información que se memorizará durante toda la visita de un usuario a una página web. Dicho de otra forma, un usuario puede ver varias páginas durante su paso por un sitio web y con sesiones podemos almacenar variables que podremos acceder en cualquiera de esas páginas.

Digamos que las sesiones son una manera de guardar información, específica para cada usuario, durante toda su visita. Cada usuario que entra en un sitio abre una sesión, que es independiente de la sesión de otros usuarios. En la sesión de un usuario podemos almacenar todo tipo de datos, como su nombre, productos de un hipotético carrito de la compra, preferencias de visualización o trabajo, páginas por las que ha pasado, etc. Todas estas informaciones se guardan en lo que denominamos variables de sesión.

PHP dispone de un método bastante cómodo de guardar datos en variables de sesión, y de un juego de funciones para el trabajo con sesiones y variables de sesión. Lo veremos en este artículo.

Para cada usuario PHP internamente genera un identificador de sesión único, que sirve para saber las variables de sesión que pertenecen a cada usuario. Para conservar el identificador de sesión durante toda la visita de un usuario a una página PHP almacena la variable de sesión en una cookie, o bien la propaga a través de la URL. Esto se puede configurar desde el archivo php.ini.

### 7.7.1.- Trabajo con sesiones en PHP

Cuando queremos utilizar variables de sesión en una página tenemos que iniciar la sesión con la siguiente función:

#### **session\_start ()**

Inicia una sesión para el usuario o continúa la sesión que pudiera tener abierta en otras páginas. Al hacer session\_start() PHP internamente recibe el identificador de sesión almacenado en la cookie o el que se envíe a través de la URL. Si no existe tal identificador se sesión, simplemente lo crea.

**Nota:** Si en el php.ini se ha definido la variable session.auto\_start = 1 se inicializa automáticamente la sesión en cada página que visita un usuario, sin que se tenga que hacer el session\_start()

Una vez inicializada la sesión con session\_start() podemos a partir de ahora utilizar variables de sesión, es decir, almacenar datos para ese usuario, que se conserven durante toda su visita o recuperar datos almacenados en páginas que haya podido visitar.

La sesión se tiene que inicializar antes de escribir cualquier texto en la página. Esto es importante y de no hacerlo así corremos el riesgo de recibir un error, porque al iniciar la sesión se deben leer las cookies del usuario, algo que no se puede hacer si ya se han enviado las cabeceras del HTTP.

**Nota:** si se intenta abrir una sesión después de haber enviado texto de la página al cliente se obtendrá el siguiente mensaje:  
Warning: session\_start(): Cannot send session cache limiter - headers already sent (output started at ...)

Una vez iniciada la sesión podemos utilizar variables de sesión a través de \$\_SESSION, que es un array asociativo, donde se accede a cada variable a partir de su nombre, de este modo:

```
$_SESSION["nombre_de_variable"]
```

**Nota:** `$_SESSION` es una variable global que existe a partir de PHP 4.1.0. Lo normal es que podamos acceder a esa variable normalmente, pero si nuestra versión de PHP no está actualizada podemos intentarlo con `$HTTP_SESSION_VARS`, que es también un array asociativo, aunque no es de ámbito global. Si `$HTTP_SESSION_VARS` tampoco funciona tendremos que registrar cada variable por separado con la función `session_register()`, enviando por parámetro los nombres de las variables de sesión que desea utilizar desde PHP.

Existen otras dos configuraciones del `php.ini` que afectan al trabajo con variables de sesión, que son `track_vars` y `register_globals`. Por defecto `track_vars` está activado y `register_globals` está desactivado. Este es el caso normal y el que suponemos tendrá el servidor donde programes, pero si esas variables cambian podría cambiar alguna cosita, como que las variables se tengan que registrar explícitamente con `session_register()`.

Ejemplo de código para definir una variable de sesión:

```
<?
session_start();
?>
<html>
<head>
<title>Generar variable de sesión</title>
</head>
<body>
<?
$_SESSION["mivariabledesesion"] = "Hola este es el valor de la variable de sesión";
?>
</body>
</html>
```

Como se puede ver, es importante inicializar la sesión antes de hacer otra cosa en la página. Luego podremos definir variables de sesión en cualquier lugar del código PHP de la página.

Para leer una variable de sesión se hace a través del mismo array asociativo `$_SESSION`. Es tan sencillo como haríamos para utilizar cualquier otra variable, lo único es que tenemos que haber inicializado la sesión previamente. Y por supuesto, que la variable que deseamos acceder exista previamente.

```
<?
session_start();
?>
<html>
<head>
<title>Leo variable de sesión</title>
</head>
<body>
Muestro esa variable:
<?
echo $_SESSION["mivariabledesesion"];
?>
</body>
</html>
```

Como se puede ver, al inicio del código hemos inicializado la sesión y luego en cualquier parte del código podríamos acceder a las variables de sesión que tuviésemos creadas.

**Nota:** si intentamos acceder a una variable de sesión con `$_SESSION` que no ha sido creada obtendremos otro mensaje de error: Notice: Undefined index: mivariabledesesion, que es el mismo que si intentamos acceder a cualquier elemento de un array que no existe.

## Más sobre sesiones en PHP

La siguiente información sobre sesiones de PHP también puede ser de útil lectura. No obstante lo expresado hasta aquí es una información mucho más actualizada. En las próximas líneas se explican mecanismos para sesiones pero todos los que se comentan, aunque son válidos, no son actuales por tratarse de explicaciones para versiones de PHP más antiguas.

Hemos dicho en el capítulo anterior que las variables de sesión se diferencian de las variables clásicas en que éstas residen en el servidor, son específicas de un solo usuario definido por un identificador y pueden ser utilizadas en la globalidad de nuestras páginas.

Para iniciar una sesión podemos hacerlo de dos formas distintas:

-Declaramos abiertamente la apertura de sesión por medio de la función `session_start()`. Esta función crea una nueva sesión para un nuevo visitante o bien recupera la que está siendo llevada a cabo.

-Declaramos una variable de sesión por medio de la función `session_register('variable')`. Esta función, además de crear o recuperar la sesión para la página en la que se incluye también sirve para introducir una nueva variable de tipo sesión.

Las sesiones han de ser iniciadas al principio de nuestro script. Antes de abrir cualquier etiqueta o de imprimir cualquier cosa. En caso contrario recibiremos un error.

Con lo visto, vamos a proponer el ejemplo clásico de utilización de una sesión: un contador. Este contador deberá aumentar de una unidad cada vez que recargamos la página o apretamos al enlace:

```
<?
session_register('contador');
?>
<HTML>
<HEAD>
<TITLE>contador.php</TITLE>
</HEAD>
<BODY>
<?
If (isset($contador)==0)
{$contador=0;}
++$contador;
echo "<a href='\"contador.php\"'>Has recargado esta página $contador veces</a>";
?>
</BODY>
</HTML>
```

### Ejecutar script

La condición *if* tiene en cuenta la posibilidad de que la variable `$contador` no haya sido todavía inicializada. La función `isset` se encarga de dar un valor cero cuando una variable no ha sido inicializada.

Otras funciones útiles para la gestión de sesiones son:

Función	Descripción
<code>Session_id()</code>	Nos devuelve el identificador de la sesión
<code>Session_destroy()</code>	Da por abandonada la sesión eliminando variables e identificador.
<code>Session_unregister('variable')</code>	Abandona una variable sesión

Para aprender más sobre las sesiones, concretamente para ver una aplicación que gestiona un carrito de compra por medio de variables sesión visita nuestro [artículo del taller de PHP](#).

Si buscas más funciones o información adicional sobre las sesiones, consulta el [manual oficial de PHP](#).

Artículo por *Rubén Álvarez*

## Parte 8:

# Bases de datos en PHP

Entramos en una de las partes más interesantes del manual de PHP, con los capítulos que tratan las bases de datos. Trabajaremos con MySQL y mostraremos cómo hacer todas las operaciones típicas con una base de datos, como acceso a registros, inserción, modificación y borrado.

### 8.1.- Trabajar con bases de datos en PHP

*Interés del empleo de bases de datos con páginas dinámicas. Presentación del lenguaje SQL y de la base MySQL. Pasos previos a los ejemplos.*

Una de las principales ventajas que presenta el trabajar con páginas dinámicas es el poder almacenar los contenidos en bases de datos. De esta forma, podemos organizarlos, actualizarlos y buscarlos de una manera mucho más simple.

El lenguaje PHP, ya hemos dicho, ofrece interfaces para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft, a partir de las cuales podremos editar el contenido de nuestro sitio con absoluta sencillez.

Esta interacción se realiza, por un lado, a partir de las funciones que PHP nos propone para cada tipo de base de datos y, por otro estableciendo un diálogo a partir de un idioma universal: SQL (Structured Query Language) el cual es común a todas las bases de datos. Este lenguaje resulta, como veremos en el [tutorial de SQL](#), muy potente y fácil de aprender.

En este manual de PHP nos limitaremos pues a la utilización las instrucciones SQL básicas que serán aprendidas a medida que explicamos las diferentes formas de actuar sobre una base de datos dejando para el [tutorial de SQL](#) los aspectos más avanzados.

Como base ejemplo de estos capítulos hemos elegido MySQL, sin duda la base de datos más extendida en combinación con PHP. Su gratuidad, eficiencia y simplicidad la han hecho una buena candidata.

Ya hemos explicado en capítulos anteriores su [instalación](#) a la vez que hemos presentado los comandos de base que nos pueden permitir abordarla con una relativa facilidad.

En caso de utilizar cualquier otra base compatible, las correcciones a llevar a cabo con respecto a nuestros ejemplos no son excesivamente grandes y la lectura de esos capítulos sigue siendo de gran utilidad.

Una vez instalado MySQL y antes de poder comenzar con nuestros ejemplos, será necesario llevar a cabo las siguientes operaciones:

-Introducidos dentro de MySQL, crearemos la base de datos ejemplo con la siguiente sentencia:

```
create database ejemplo;
```

-Seleccionaremos la base ejemplo como la base a utilizar:

```
use ejemplo
```

-Crearemos a continuación la tabla clientes a partir de la siguiente sentencia:

```
create table clientes (  
nombre varchar(100),  
telefono varchar(100)  
);
```

Ahora ya disponemos de nuestra tabla vacía. Sólo queda comenzar a llenarla con los datos que iremos insertando.

El conjunto de scripts utilizados para el bloque de bases de datos puede ser descargado [aquí](#).

Además contamos con un [videotutorial sobre las bases de datos en PHP](#) que te ayudará a entenderlo todo mejor.

Artículo por *Rubén Álvarez*

## 8.2.- Introducción de nuevos registros con PHP

*Forma sencilla de introducir nuevos elementos en una tabla de base de datos. Realizamos un Insert con PHP.*

Una vez creada la tabla *clientes* en nuestra base de datos *ejemplo*, el paso siguiente sea llenarla con registros. Para ello vamos a ver este artículo, en el que se reciben datos desde un formulario y luego se insertan con PHP en la base de datos, en la tabla adecuada.

Los datos del registro pueden ser recogidos, por ejemplo, a partir de un formulario. Aquí os proponemos un simple documento HTML que recoge los datos y los envía a una página PHP que se encarga de procesarlos:

```
<HTML>
<HEAD>
<TITLE>Insertar.html</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Insertar un registro</h1>
<br>
<FORM METHOD="POST" ACTION="insertar.php">
Nombre<br>
<INPUT TYPE="TEXT" NAME="nombre"><br>
Teléfono<br>
<INPUT TYPE="TEXT" NAME="telefono"><br>
<INPUT TYPE="SUBMIT" value="Insertar">
</FORM>
</div>
</BODY>
</HTML>
```

Llegados a la página destino del formulario (*insertar.php*), lo primero que habrá que hacer es establecer un vínculo entre el programa y la base de datos. Esta conexión se lleva a cabo con la función *mysql\_connect*. A continuación, deberemos generar una orden de inserción del registro en lenguaje SQL. Esta orden será ejecutada por medio de la función *mysql\_db\_query*. En esta función especificaremos primeramente la base de datos sobre la que queremos actuar y a continuación introduciremos la sentencia SQL:

```
<HTML>
<HEAD>
<TITLE>Insertar.php</TITLE>
</HEAD>
<BODY>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//selección de la base de datos con la que vamos a trabajar
mysql_select_db("mi_base_datos");

//Ejecucion de la sentencia SQL
mysql_query("insert into clientes (nombre,telefono) values ('$nombre','$telefono')");
?>
<h1><div align="center">Registro Insertado</div></h1>
<div align="center"><a href="lectura.php">Visualizar el contenido de la base</a></div>
</BODY>
```



&lt;/HTML&gt;

### Ejecutar ejemplo

Los parametros user y password son definidos por el creador de la base. Es conveniente en un principio, al crear nuestras bases, trabajar sin ellos con lo cual dejaremos las cadenas correspondientes vacias: "".

Además de la propia inserción, el programa avisa de la introducción del registro y ofrece un enlace hacia una página de lectura la cual será comentada a continuación.

No entraremos en la descripción de la orden SQL, para comprender más acerca de cómo introducir registros, refererirse al [tutorial de SQL](#).

Artículo por *Rubén Álvarez*

## 8.3.- Selección y lectura de registros con PHP

*Utilizamos el comando Select de SQL para crear una selección de nuestra tabla y mostrar todos los datos en pantalla por medio de un bucle. Con PHP.*

Dentro de una base de datos, organizada por tablas, la selección de una tabla entera o de un cierto numero de registros resulta una operación rutinaria.

Aquí os mostramos una forma bastante clásica de mostrar en pantalla a partir de un bucle los registros seleccionados por una sentencia SQL:

```
<HTML>
<HEAD>
<TITLE>lectura.php</TITLE>
</HEAD>
<BODY>
<h1><div align="center">Lectura de la tabla</div></h1>
<br>
<br>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//selección de la base de datos con la que vamos a trabajar
mysql_select_db("mi_base_datos");

//Ejecutamos la sentencia SQL
$result=mysql_query("select * from clientes");
?>
<table align="center">
<tr>
<th>Nombre</th>
<th>Teléfono</th>
</tr>
<?
//Mostramos los registros
while ($row=mysql_fetch_array($result))
{
echo '<tr><td>'.$row["nombre"].'</td>';
echo '<td>'.$row["telefono"].'</td></tr>';
}
mysql_free_result($result)
?>
</table>

<div align="center">
<a href="insertar.html">Añadir un nuevo registro</a><br>
<a href="actualizar1.php">Actualizar un registro existente</a><br>
<a href="borrar1.php">Borrar un registro</a><br>
```

```
</div>
</BODY>
</HTML>
```

### Ejecutar script

Los pasos a realizar son, en un principio, los vistos para la inserción de un registro: Conexión a la base y ejecución de la sentencia. Esta vez, la información de dicha ejecución será almacenada en una variable (*\$result*).

El siguiente paso será plasmar en pantalla la información recogida en *\$result*. Esto lo haremos mediante la función *mysql\_fetch\_array* que devuelve una variable array con los contenidos de un registro a la vez que se posiciona sobre el siguiente. El bucle *while* nos permite leer e imprimir secuencialmente cada uno de los registros.

La función *mysql\_free\_result* se encarga de liberar la memoria utilizada para llevar a cabo la consulta. Aunque no es necesaria su utilización, resulta altamente aconsejable.

Artículo por *Rubén Álvarez*

## 8.4.- Actualización de un registro de base de datos con PHP

*Explicamos cómo modificar o actualizar un registro existente en una tabla de una base de datos, con PHP. Es decir, hacer un update para un registro de una tabla, desde PHP.*

Para mostrar cómo se actualiza un registro presente en nuestra base de datos, vamos a hacerlo a partir de un caso un poco más complejo para que empecemos a familiarizarnos con estas operaciones. Realizaremos un par de scripts que permitan cambiar el número de teléfono de las distintas personas presentes en nuestra base. El nombre de estas personas, así como el nuevo número de teléfono, serán recogidos por medio de un formulario.

El archivo del formulario va a ser esta vez un script PHP en el que efectuaremos una llamada a nuestra base de datos para construir un menú desplegable donde aparezcan todos los nombres. La cosa quedaría así:

```
<HTML>
<HEAD>
<TITLE>Actualizar1.php</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Actualizar un registro</h1>
<br>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//selección de la base de datos con la que vamos a trabajar
mysql_select_db("mi_base_datos");

echo '<FORM METHOD="POST" ACTION="actualizar2.php">Nombre<br>';

//Creamos la sentencia SQL y la ejecutamos
$sql="Select nombre From clientes Order By nombre";
$result=mysql_query($sql);

echo '<select name="nombre">';

//Generamos el menu desplegable
while ($row=mysql_fetch_array($result))
{echo '<option>'.$row["nombre"];}
?>
</select>
<br>
Teléfono<br>
<INPUT TYPE="TEXT" NAME="telefono"><br>
```

```
<INPUT TYPE="SUBMIT" value="Actualizar">
</FORM>
</div>
```

```
</BODY>
</HTML>
```

La manera de operar para construir el menú desplegable es la misma que para visualizar la tabla. De nuevo empleamos un bucle *while* en combinación con la función *mysql\_fetch\_array* lo que nos permite mostrar cada una de las opciones.

El script de actualización será muy parecido al de inserción:

```
<HTML>
<HEAD>
<TITLE>Actualizar2.php</TITLE>
</HEAD>
<BODY>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//selección de la base de datos con la que vamos a trabajar
mysql_select_db("mi_base_datos");

//Creamos la sentencia SQL y la ejecutamos
$sql="Update Clientes Set telefono='$telefono' Where nombre='$nombre'";
mysql_query($sql);
?>

<h1><div align="center">Registro Actualizado</div></h1>
<div align="center"><a href="lectura.php">Visualizar el contenido de la base</a></div>

</BODY>
</HTML>
```

### Ejecutar ejemplo

Artículo por *Rubén Álvarez*

## 8.5.- Borrado de un registro con PHP

*Aprendemos a eliminar registros de una tabla de base de datos, con un ejemplo práctico en PHP.*

Otra de las operaciones elementales que se pueden realizar sobre una base de datos es borrar un registro. Para hacerlo, SQL nos propone sentencias del tipo *Delete*. Veámoslo con un ejemplo aplicado a nuestra agenda.

Cabe señalar que primero debemos seleccionar el registro que se desea borrar y luego realizar el borrado propiamente dicho. Para ello crearemos un menú desplegable dinámico, donde se podrá seleccionar el elemento que se desea borrar. Luego se pasará a una página PHP una referencia al elemento seleccionado, para borrarlo de la base de datos.

```
<HTML>
<HEAD>
<TITLE>Borrar1.php</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Borrar un registro</h1>
<br>

<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//selección de la base de datos con la que vamos a trabajar
mysql_select_db("mi_base_datos");
```

```
echo '<FORM METHOD="POST" ACTION="borrar2.php">Nombre<br>';

//Creamos la sentencia SQL y la ejecutamos
$$SQL="Select nombre From clientes Order By nombre";
$result=mysql_query($$SQL);

echo '<select name="nombre">';

//Mostramos los registros en forma de menú desplegable
while ($row=mysql_fetch_array($result))
{echo '<option>'. $row["nombre"];}
mysql_free_result($result)
?>

</select>
<br>
<INPUT TYPE="SUBMIT" value="Borrar">
</FORM>
</div>

</BODY>
</HTML>
```

El siguiente paso es hacer efectiva la operación a partir de la ejecución de la sentencia SQL que construimos a partir de los datos del formulario:

```
<HTML>
<HEAD>
<TITLE>Borrar2.php</TITLE>
</HEAD>
<BODY>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//selección de la base de datos con la que vamos a trabajar
mysql_select_db("mi_base_datos");

//Creamos la sentencia SQL y la ejecutamos
$$SQL="Delete From Clientes Where nombre='$nombre'";
mysql_query($$SQL);
?>

<h1><div align="center">Registro Borrado</div></h1>
<div align="center"><a href="lectura.php">Visualizar el contenido de la base</a></div>

</BODY>
</HTML>
```

### Ejecutar ejemplo

Con este capítulo cerramos el bloque de accesos a bases de datos con PHP. Para mas información relacionada podéis referiros al [taller de PHP](#) donde podréis encontrar algún que otro artículo interesante al respecto.

*Artículo por Rubén Álvarez*

## Parte 9:

# Subir una aplicación web al servidor

Una vez que hemos terminado una aplicación web en local, tenemos que ponerla en producción en un servidor de Internet. Mostramos cómo subir todas las páginas a un servidor y algunas de las posibles tareas que nos tocará realizar para subir también la base de datos.

### 9.1.- Subir una aplicación PHP al servidor

*Vamos a ver cómo subir una aplicación hecha en local a un servidor de Internet. Empezamos ofreciendo una serie de pautas para subir los archivos.*

En el pasado me solicitaron que escribiese sobre un tema que hasta ahora no habíamos tocado más que de refilón, que consiste en la puesta en marcha de una aplicación, programada en local, a nuestro servidor de hosting, es decir, en el paso de subir todos los archivos PHP y la base de datos a nuestro espacio en el servidor web contratado en un proveedor de alojamiento.

El tema espero que resulte familiar a muchas de las personas que leen nuestros artículos, ya que probablemente hayan tenido que pasar por esa etapa en alguna ocasión, aunque pretendo dar algunas claves y truquillos que pueden ayudar a todos, tengan o no experiencia en este asunto.

#### 9.1.1.- Subir los archivos

Nuestro servidor web debe tener un directorio para la publicación de las páginas web. Ese sería el lugar donde hay que subir los archivos .php.

Dependiendo del proveedor con el que trabajemos, el directorio de publicación puede variar. Generalmente, cuando contratamos un alojamiento, nos proporcionan una cuenta de FTP con la que conectarnos al servidor web y transferir los archivos de nuestro sitio, además de unos datos para la conexión, que serán el nombre del servidor y el usuario y contraseña para el acceso al FTP.

**Referencia:** por si alguien no sabe lo que es el FTP, hablamos más sobre ello en el manual de [Publicar en Internet](#), concretamente en el artículo [Subir los archivos al servidor](#).

Al conectarnos al servidor con los datos del FTP, que deben ser proporcionados por nuestro proveedor, accederemos a un directorio. Este directorio podría ser el de publicación, aunque generalmente no es así, sino que suele ser un subdirectorio llamado "HTML" o "docs" o algo similar, que cuelga del directorio de inicio en nuestra conexión FTP. Como decía, este directorio puede tener nombres distintos en proveedores distintos, aunque, en cualquier caso, con una simple pregunta a nuestro proveedor resolveremos esa duda.

Los archivos se deben subir al directorio de publicación, o a cualquier subdirectorio de este. En definitiva, los tendremos que alojar por ahí dentro y para acceder a ellos bastaría con escribir el nombre del dominio o URL de nuestro alojamiento,

seguido del nombre del archivo. Si tuviésemos un archivo llamado hola.php y nuestro alojamiento se ha contratado para el dominio [www.midominio.com](http://www.midominio.com), deberíamos subir ese archivo al directorio de publicación y accederíamos al archivo escribiendo:

<http://www.midominio.com/hola.php>

Si creamos subdirectorios dentro del directorio de publicación podremos acceder a ellos escribiendo el nombre del dominio o URL de nuestro alojamiento, seguido del nombre del directorio y el nombre del archivo. Por ejemplo, si creamos un subdirectorio llamado paginas y tenemos dentro un archivo llamado pag1.php, podríamos acceder a él de la siguiente manera.

<http://www.midominio.com/paginas/pag1.php>

**Referencia:** hay otro concepto interesante que deberíamos conocer llegados a este punto, que es el "documento por defecto". Éste no es más que el archivo que se envía al navegador si en la URL accedida no se especificaba ningún archivo. Suele llamarse `index.html` o `index.php` (o `index.asp` si nuestro servidor soporta programación en ASP), aunque puede variar de un proveedor a otro. Hablamos más sobre el [documento por defecto](#) en nuestro manual de Publicar en Internet.

Artículo por Miguel Angel Alvarez

## 9.2.- Colocar los archivos PHP fuera del directorio de publicación

*Algunos casos en los que colocar archivos fuera del directorio de publicación tiene sentido y utilidad.*

Por decir algo más sobre el tema de colocar los archivos, quería señalar que cualquier cosa que pongamos fuera del directorio de publicación no podrá ser accedida a través del navegador. Es decir, si creamos un directorio que se llame `funciones_php` en el mismo nivel que el directorio de publicación (fuera del directorio de publicación) no podremos acceder con el explorador a los archivos que coloquemos dentro de ninguna de las maneras. Esto es así porque la URL de inicio de nuestro alojamiento corresponde con ese directorio y no podemos movernos hacia debajo de ese directorio con las URLs, que son la manera de especificar al navegador los recursos a los que se quiere acceder.

**Nota:** Ya se explicó lo que era el directorio de publicación en el capítulo anterior sobre [Subir archivos PHP al servidor](#).

No sería posible salir del directorio de publicación con una URL como esta, por mucho que utilicemos el operador `..` (que sirve para acceder al directorio padre).

[http://www.midominio.com/./funciones\\_php/archivo\\_inalcanzable.php](http://www.midominio.com/./funciones_php/archivo_inalcanzable.php)

Sin embargo, colocar algunos contenidos fuera del directorio de publicación puede ser muy útil. Por ejemplo, podríamos colocar allí copias de seguridad de algunos archivos o documentos que simplemente queremos guardar en el servidor para acceder a ellos desde cualquier parte y con nuestro programa de FTP.

Hay otra utilidad más interesante sobre colocar archivos fuera del directorio de publicación. Se trata de que muchas veces utilizamos en nuestros programas trozos de código repetidamente, por ejemplo, para abrir y cerrar bases de datos, para mostrar la cabecera de nuestro portal, para comprobar que un email escrito en un formulario es correcto, etc. Es muy útil separar estos trozos de código en un archivo a parte y llamar a este archivo con las funciones PHP `include()` o `require()`. Así, si un día modificamos la cabecera de nuestro portal, sólo lo tendremos que modificar en un archivo, o, si cambia la base de datos que utilizamos sólo tendríamos que modificar el archivo que hace la conexión a la base de datos una vez, en lugar de ir cambiándolo en todas las páginas PHP que abrían las bases de datos.

Estos archivos no son páginas independientes, sino trozos. Seguramente, si los ejecutamos por separado no mostrarían ningún resultado válido, incluso podrían dar mensajes de error. Por esta razón merece la pena colocarlos en un lugar donde nadie pueda tener acceso: fuera del directorio de publicación. Con PHP sí que podremos acceder a ese directorio para incluir esos archivos. Solamente deberíamos utilizar las funciones PHP `include()` o `require()` indicando la ruta para acceder a los

archivos.

En el caso de que tengamos una página llamada hola.php en el directorio de publicación y un archivo, que se llama abre\_base\_datos.php, en el directorio funciones\_php, que está fuera del directorio de publicación, si quisiéramos acceder (desde hola.php) al archivo que abre la base de datos lo haríamos así.

```
include("../funciones_php/abre_base_datos.php")
```

Desde PHP sí que podemos acceder a los archivos que se encuentran fuera del directorio de publicación. Para ello especificamos la ruta adecuada, en la que utilizamos el operador .. para bajar al directorio padre.

Nada más que decir sobre la colocación de los archivos: una vez situados en el directorio de publicación se podrá acceder a ellos con nuestro navegador y se deberían ejecutar perfectamente. Aunque cabe señalar que, tanto PHP como el servidor donde trabajemos, pueden tener configuraciones distintas y puede que algún detalle de la programación de nuestras páginas no funcione correctamente.

Por ejemplo, nuestro PHP puede declarar o no automáticamente las variables que llegan a través de un formulario. Si en local sí que estaba configurado para hacer esto y en remoto no, deberíamos localizar los lugares donde recogemos las variables y utilizar las variables de entorno correctas (mirar artículo sobre [Procesar variables de formularios](#) y los comentarios al pie para saber más de esta posible fuente de errores).

Aunque este no es un caso habitual, podemos ponernos en contacto con nuestro proveedor de alojamiento para ver si pueden ayudarnos configurando el sistema o indicando los pasos a seguir para solventar en nuestros scripts el asunto.

**Comentario:** el siguiente script calcula el nivel de directorio de los scripts que queremos incluir en la página actual.

```
// Hallamos el nivel de directorio en que está ubicada la página
1. Se hace un recuento de los caracteres que contiene el nombre del script
actual.
-
$Chars = count_chars($PHP_SELF,1);
-
2. Exploramos la tabla de los caracteres devueltos buscando el carácter ('/'
Código 47 ) de directorio (carpeta) que devuelve Apache.
-
foreach ($Chars as $Char=>$nChars){
    if ($Char==47) {$n=$nChars;break;}
}
-
3. Generamos una cadena de n-1 veces con la subcadena "../" que nos da el
nivel de directorio en que se encuentra el script.
-
if ($n==0) $PathString=""; else $PathString=str_pad("", ($n-1)*3, "../");
```

Artículo por *Miguel Ángel Álvarez*

## 9.3.- Subir una base de datos al servidor de Internet

*El segundo paso para subir una aplicación PHP al servidor consiste en colocar la base de datos en el*

Aparte de los archivos de la página, debemos subir la base de datos con la que tenemos que trabajar. Las bases de datos con las que trabaja PHP son muy variadas y en distintos casos podemos utilizar una u otra, por lo que los modos de subir la base de datos también pueden variar.

**Nota:** Este artículo y los sucesivos, que tratan sobre subir una base de datos MySQL al servidor, se engloban tanto dentro del [Manual de PHP](#) como del [Taller de MySQL](#). Por ello, será importante disponer de conocimientos de ambas tecnologías para entender y aprovechar estas explicaciones.

Es muy corriente que nuestro proveedor de hosting ofrezca junto con PHP la base de datos MySQL, así que las notas para subir esa base de datos al servidor de este artículo van encaminadas a ofrecer soluciones para esa base de datos.

La base de datos MySQL no se puede subir por FTP, como que se hacía con los archivos del código PHP. Para subirla tendremos que utilizar otros mecanismos. Voy a distinguir entre tres casos distintos en los que nos podríamos encontrar en este momento:

1. **La base de datos que pretendemos subir está vacía.** Tan sólo hemos creado las tablas, pero no hemos introducido datos en ellas o, a lo sumo, tienen algún dato que hemos introducido de pruebas.
2. **La base de datos que queremos subir está completa y es una base de datos MySQL.** En este caso tenemos creada la base de datos en local y con toda la información dentro y, por supuesto, queremos que esa información quede también en la base de datos remota.
3. La base de datos está **completa** (como el caso anterior), pero **no es una base de datos MySQL**. En este caso estaríamos haciendo una migración de la base de datos de un sistema gestor a otro.

Veremos los tres casos por separado en adelante, aunque, antes de ello, vamos a mostrar unas herramientas que nos servirán de mucha ayuda para la administración de cualquier base de datos remota.

Las herramientas en concreto se relatan en el manual [Taller de MySQL](#), son las siguientes:

- **PhpMyAdmin.** Una aplicación creada en PHP que podemos instalar en nuestro espacio de alojamiento para administrar la base de datos.
- **MySql Control Center** (en adelante MyCC). Una aplicación Windows que permite conectarse a múltiples bases de datos MySQL, que se encuentren en local o en remoto.
- **Access.** También permite administrar una base de datos MySQL conectada en local o en remoto. En este caso se utiliza una interfaz que muchos ya conocen, como es Access, para administrar una base de datos que nada tiene que ver con dicho programa.

En los tres casos lo que nos permite realizar el software de administración son tareas sobre la base de datos de todo tipo, como pueden ser crear tablas, modificarlas, insertar datos, borrarlos, editarlos. Modificar o borrar tablas o campos de las mismas, etc.

La elección de una herramienta o de otra pasa por los recursos que nos permitan utilizar en nuestro proveedor. Básicamente, lo que nos puede decantar a una opción u otra, es si permiten o no conectar de manera remota la base de datos MySQL. Conozco alojamientos donde se permite esa conexión remota y donde no.

Si no permiten conectarnos remotamente nos decantaremos por PhpMyAdmin, pues es una aplicación PHP que se conecta en local y a la que se accede desde una página web y eso lo permiten todos los proveedores, incluso hay muchos que tienen instalado ya este software para administrar las bases de datos.

En caso de que sí nos permitan conectarnos remotamente con la base de datos, eliremos MyCC o Access, que son aplicaciones Windows mucho más potentes y rápidas que las que utilizan interfaz web, como PhpMyAdmin. Es preferible utilizar MyCC porque está especialmente desarrollado para conectar y operar con bases de datos MySQL.

*Artículo por Miguel Ángel Álvarez*

## 9.4.- Subir base de datos MySQL vacía al servidor

*La base de datos que pretendemos subir está vacía. Tan sólo hemos creado las tablas, pero no hemos introducido datos en ellas o, a lo sumo, tienen algún dato que hemos introducido de pruebas.*

Es muy normal que hayamos diseñado una base de datos para nuestro proyecto desde 0, definiendo las distintas entidades de nuestro modelo de datos, junto con sus campos y sus tipos.

En estos casos lo más probable es que la base de datos esté vacía, o bien contenga datos que hayamos introducido a modo de prueba y que no queramos conservar cuando subamos la aplicación a Internet.

La opción más interesante entonces podría ser crear otra vez las tablas que tenemos en local en la base de datos remota. Para



ello tenemos dos posibilidades.

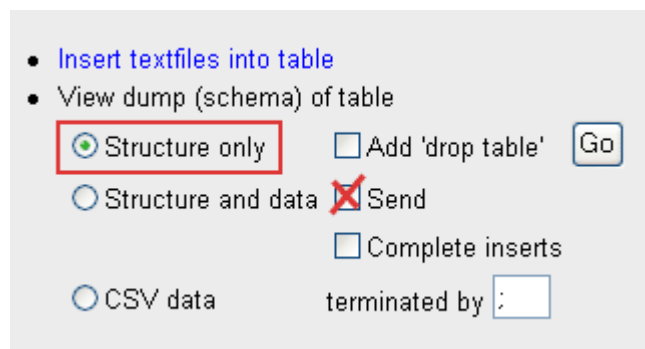
**a) Si tenemos pocas tablas y bastante sencillas**

Las podemos crear en remoto con alguna herramienta como [PhpMyAdmin](#) o [MyCC](#).

**b) Si tiene muchas tablas y/o muy complicadas**

La recomendación sería hacer un backup de la estructura en local y restaurarla en remoto. Esto nos evitará tener que volver a crear todas las tablas y definir todos sus campos y sus tipos. Puede ser un poco más complicado pero sin duda nos ahorrará tiempo.

Para hacer el backup de la estructura en local podemos utilizar alguna herramienta como [PhpMyAdmin](#), o bien utilizar el comando [mysqldump](#) desde línea de comandos de MS-DOS.



Herramienta de backup de PhpMyAdmin. Está marcada la opción de extraer solamente la estructura de las tablas. Si marcamos además la casilla "Send", nuestro navegador se descargará el backup en un fichero de texto. Si no lo pulsamos simplemente se visualizará.

Lo que tenemos que hacer en este caso es un backup de la estructura de la base de datos, es decir, los "create tables" o sentencias SQL para crear las tablas. Sería un montón de sentencias con esta forma:

```
# -----
#
# Table structure for table 'comentario'
#
CREATE TABLE comentario (
  id_comentario int(5) unsigned NOT NULL auto_increment,
  id_articulo int(4) DEFAULT '0' NOT NULL,
  comentario text NOT NULL,
  fecha int(14) unsigned DEFAULT '0' NOT NULL,
  revisado tinyint(1) DEFAULT '0' NOT NULL,
  nombre_comentario varchar(100) DEFAULT 'Nombre no especificado' NOT NULL,
  email_comentario varchar(100) DEFAULT 'Email sin especificar' NOT NULL,
  tipo tinyint(1) unsigned DEFAULT '1' NOT NULL,
  PRIMARY KEY (id_comentario)
);
```

Para restaurar estas sentencias tenemos opciones tanto dentro de PhpMyAdmin como de MyCC. En ambos casos lo que tenemos que hacer es ejecutar estas sentencias en el servidor MySQL remoto. En PhpMyAdmin tenemos un campo para introducir sentencias SQL y también otro campo para seleccionar un archivo de texto con todas las sentencias SQL, para ejecutarlas una detrás de otra. En MyCC tenemos un botón que nos permite abrir una consola donde introducir una o varias sentencias SQL y ejecutarlas.

### Herramienta de backup y restauración de PhpMyAdmin

Para restaurar la tabla desde el backup  
compuesto por sentencias SQL

- [Print view](#)
- Run SQL query/queries on database [www.desarrolloweb.com](#) ([Documentation](#)):

or Location of the textfile:

**Introducir una o varias sentencias SQL**

**Indicar un fichero con sentencias SQL**

---

- [Query by Example](#)
- View dump (schema) of database

☒ Structure only

☐ Structure and data
 ☐ Send
 ☐ Complete insert

**Backup de la estructura de las tablas**

**La estructura y los datos de las tablas**

Para obtener el backup  
de la base de datos

Botón para introducir sentencias SQL en MyCC



Repetimos, esto sólo nos servirá para subir la estructura de la base de datos y no los datos que contenga. Si deseamos subir también la información de la base de datos entonces debemos utilizar otras estrategias, relatadas próximamente.

Artículo por *Miguel Ángel Álvarez*

## 9.5.- Subir una base de datos MySQL con la estructura y los datos

*Como transferir una base de datos MySQL que tenemos en local a nuestro servidor remoto, incluyendo tanto la estructura de las tablas como sus datos.*

Si la base de datos que deseamos subir está llena de información y deseamos que se conserve una vez subida la base de datos a remoto, tenemos que realizar un backup de la base de datos y restaurarlo en remoto.

**Nota:** Estas recomendaciones están pensadas para subir una base de datos MySQL que podamos tener en local a una base de datos MySQL que hayamos contratado en remoto. Si la base origen no es MySQL estaríamos hablando de una migración de bases de datos, pero esto lo veremos en un [artículo más adelante](#).

En este caso el procedimiento sería muy parecido al de [subir una base de datos vacía](#), relatado anteriormente, con la salvedad de que ahora debemos extraer no solo la estructura de la base de datos, sino también los registros que contiene.

Para ello podemos utilizar mysqldump, según se relata en [este artículo](#), o bien [PhpMyAdmin](#), seleccionando la opción que indica que el backup contenga la estructura y los datos (Structure and data en versiones en inglés).

La estructura y los datos vendrán en un fichero de texto con una serie de sentencias SQL para crear las tablas y los insert necesarios para introducir cada uno de los datos.

Para restaurar la base de datos lo haremos tal como se ha relatado para el caso de que la base de datos estuviera vacía, con la ayuda de una instalación de PhpMyAdmin en remoto o un MyCC que se conecte a la base de datos contratada en el servidor de Internet.

Si tenemos problemas para subir el fichero de backup de la base de datos es posible que en nuestro proveedor de alojamiento nos pueda ayudar a subir el fichero y restaurarlo. Como el proveedor dispone de los servidores en sus propias instalaciones, tiene muchas más posibilidades que nosotros para trabajar con las bases de datos, sin temor a que las lentas comunicaciones por Internet arrojen errores en la restauración de los datos.

Si nuestro proveedor no puede ayudarnos, seguramente disponga y nos indique algún mecanismo para realizar la tarea sin lugar a errores. Puede ocurrirnos con algún proveedor que nos diga que se encarga de todo pero nos exija el pago de las horas de trabajo del informático que va a restaurar el backup de la base de datos. Si no pone facilidades ni siquiera en esto posiblemente sea mejor ir pidiéndoles que nos devuelvan el dinero invertido porque su servicio no sería muy bueno.

Artículo por *Miguel Angel Alvarez*

## 9.6.- Migrar una base de datos a MySQL

*Indicaciones útiles para migrar una base de datos a MySQL, es decir, cuando tenemos que subir una base de datos local en cualquier gestor a una base de datos remota en MySQL.*

El último caso en el que nos podemos encontrar a la hora de subir una base de datos a nuestro proveedor de alojamiento es que la base de datos la tengamos creada en local, pero en un sistema gestor distinto del que vamos a utilizar en remoto. En remoto suponemos siempre que vamos a utilizar la base de datos [MySQL](#). En local podríamos disponer de una base de datos [Access](#), [SQL Server](#) o de otro sistema de base de datos.

El proceso de la migración puede ser bastante complejo y, como hay tantas bases de datos distintas, difícil de dar una receta que funcione en todos los casos. Además, aparte de la dificultad de transferir la información entre los dos sistemas gestores de base de datos, también nos influirá mucho en la complejidad del problema el tipo de los datos de las tablas que estamos utilizando. Por ejemplo, las fechas, los campos numéricos con decimales o los booleanos pueden dar problemas al pasar de un sistema a otro porque pueden almacenarse de maneras distintas o, en el caso de los números, con una precisión distinta.

### 9.6.1.- Recomendaciones para migrar de Access a MySQL

Si nuestra base de datos anterior estaba construida en Access lo tenemos bastante fácil, gracias a que [MySQL dispone de un driver ODBC para sistemas Windows](#), que nos permite [conectar Access con el propio MySQL](#) y pasar información fácilmente.

Este tema está relatado en el artículo [Exportar datos de MySQL a Access](#), aunque hay que indicar que si deseamos hacer una exportación desde Access en local a MySQL en remoto puede haber problemas porque no todos los alojadores permiten las conexiones en remoto con la base de datos. Si no tenemos disponible una conexión en remoto con nuestro servidor de bases de datos vamos a tener que cambiar la estrategia un poco.

La idea en este último caso es instalar MySQL en local y realizar la migración desde Access en local a MySQL en local y luego podríamos [hacer un backup de la base de datos local y subirla a remoto](#), tal y como se ha relatado antes.

### 9.6.2.- Recomendaciones para migrar desde SQL Server a MySQL

La verdad es que no he tenido este caso nunca, pero hay que decir que Access también nos puede ayudar en este caso. Access permite seleccionar una base de datos SQL Server y trabajar desde la propia interfaz de Access. La idea es que Access también permite trabajar con MySQL y posiblemente haciendo un puente entre estos dos sistemas gestores podemos exportar datos de SQL Server a MySQL.

Lo que es seguro que utilizando el propio Access de puente podríamos realizar el trabajo. Primero exportando de SQL Server a Access y luego desde Access a MySQL.

### 9.6.3.- Otras bases de datos u otras técnicas

Si la base de datos origen dispone de un driver ODBC no habrá (en teoría) problema para conectarla con Access, de manera similar a como se conecta con MySQL. Entonces podríamos utilizar Access para exportar los datos, porque desde allí se podrían acceder a los dos sistemas gestores de bases de datos.

Si no tenemos Access, o la base de datos original no tiene driver ODBC, o bien no nos funciona correctamente el proceso y no sabemos cómo arreglarlo, otra posibilidad es exportar los datos a ficheros de texto, separados por comas o algo parecido. Muchas bases de datos tienen herramientas para exportar los datos de las tablas a ficheros de texto, los cuales se pueden luego introducir en nuestro sistema gestor destino (MySQL) con la ayuda de alguna herramienta como PhpMyAdmin.

Para ello, en la página de propiedades de la tabla encontraremos una opción para hacer el backup de la tabla y para introducir ficheros de texto dentro de una tabla (Insert textfiles into table en inglés).

- **Insert textfiles into table**
- View dump (schema) of table
  - ☒ Structure only ☐ Add 'drop table'
  - ☐ Structure and data ☐ Send
  - ☐ CSV data ☐ Complete inserts
  - terminated by

Accediendo a ese enlace podremos ver un formulario donde introducir las características del fichero de texto, como el carácter utilizado como separador de campos, o el terminador de líneas, etc, junto con el propio archivo con los datos, y PhpMyAdmin se encargará de todo el trabajo de incluir esos datos en la tabla.

Location of the textfile	<input type="text"/>	<input type="button" value="Examinar..."/>
Replace table data with file	<input type="checkbox"/> Replace	The contents of the file replaces the contents of the selected table for rows with identical primary or unique key.
Fields terminated by	<input type="text" value=";"/>	The terminator of the fields.
Fields enclosed by	<input type="text" value="\"/> <input type="checkbox"/> OPTIONALLY	Often quotation marks. OPTIONALLY means that only char and varchar fields are enclosed by the "enclosed by"-character.
Fields escaped by	<input type="text" value="\\"/>	Optional. Controls how to write or read special characters.
Lines terminated by	<input type="text" value="\n"/>	Carriage return: \r Linefeed: \n
Column names	<input type="text"/>	If you wish to load only some of a table's columns, specify a comma separated field list.
<a href="#">[Documentation]</a>		
<input type="button" value="Submit"/> <input type="button" value="Reset"/>		

Como se habrá supuesto, es necesario tener creada la tabla en remoto para que podamos introducirle los datos del fichero de texto.

#### 9.6.4.- Cambios de un formato de datos a otro

Toda la migración tiene que tener en cuenta muy especialmente, como ya se señaló, las maneras que tenga cada base de datos de guardar la información, es decir, del formato de sus tipos de datos. Tenemos que contar siempre con la posible necesidad de transformar algunos datos como pueden ser los campos booleanos, fechas, campos memo (texto con longitud indeterminada), etc, que pueden almacenarse de maneras distintas en cada uno de los sistemas gestores, origen y destino.

En algunos casos posiblemente tengamos que realizar algún script que realice los cambios necesarios en los datos. Por ejemplo puede ser para localizar los valores booleanos guardados como true / false a valores enteros 0 / 1, que es como se guarda en MySQL. También las fechas pueden sufrir cambios de formato, mientras que en Access aparecen en castellano (dd/mm/aaaa) en MySQL aparecen en el formato aaaa-mm-dd. PHP puede ayudarnos en la tarea de hacer este script, también Visual Basic Script para Access puede hacer estas tareas complejas y el propio lenguaje SQL, a base de sentencias dirigidas contra la base de datos, puede servir para algunas acciones sencillas.

Artículo por *Miguel Angel Alvarez*

## Parte 10: Introducción a la programación orientada a objetos en PHP 4

Capítulos sobre la programación orientada a objetos que se puede realizar en PHP 4. Es una primera aproximación a la orientación a objetos, tal como se realizaba en esa versión de PHP.

### 10.1.- Programación orientada a objetos en PHP

*PHP ofrece funcionalidades propias de la POO. En este capítulo veremos los aspectos más básicos de esta metodología.*

La programación orientada a objetos es una metodología de programación avanzada y bastante extendida, en la que los sistemas se modelan creando clases, que son un conjunto de datos y funcionalidades. Las clases son definiciones, a partir de las que se crean objetos. Los objetos son ejemplares de una clase determinada y como tal, disponen de los datos y funcionalidades definidos en la clase.

La programación orientada a objetos permite concebir los programas de una manera bastante intuitiva y cercana a la realidad. La tendencia es que un mayor número de lenguajes de programación adopten la programación orientada a objetos como paradigma para modelizar los sistemas. Prueba de ello es la nueva versión de PHP (5), que implanta la programación de objetos como metodología de desarrollo. También Microsoft ha dado un vuelco hacia la programación orientada a

objetos, ya que .NET dispone de varios lenguajes para programar y todos orientados a objetos.

Así pues, la programación orientada a objetos es un tema de gran interés, pues es muy utilizada y cada vez resulta más esencial para poder desarrollar en casi cualquier lenguaje moderno. En este artículo vamos ver algunas nociones sobre la programación orientada a objetos en PHP. Aunque es un tema bastante amplio, novedoso para muchos y en un principio, difícil de asimilar, vamos a tratar de explicar la sintaxis básica de PHP para utilizar objetos, sin meternos en mucha teoría de programación orientada a objetos en general.

Referencia: Se puede acceder a una descripción de la programación orientada a objetos, publicada en DesarrolloWeb.com.  
<http://www.desarrolloweb.com/articulos/499.php>

### 10.1.1.- Las clases: class

Una clase es un conjunto de variables, llamados atributos, y funciones, llamadas métodos, que trabajan sobre esas variables. Las clases son, al fin y al cabo, una definición: una especificación de propiedades y funcionalidades de elementos que van a participar en nuestros programas.

Por ejemplo, la clase "Caja" tendría como atributos características como las dimensiones, color, contenido y cosas semejantes. Las funciones o métodos que podríamos incorporar a la clase "caja" son las funcionalidades que deseamos que realice la caja, como `introduce()`, `muestra_contenido()`, `comprueba_si_cabe()`, `vaciate()`...

Las clases en PHP se definen de la siguiente manera:

```
<?
class Caja{
    var $alto;
    var $ancho;
    var $largo;
    var $contenido;
    var $color;

    function introduce($cosa){
        $this->contenido = $cosa;
    }

    function muestra_contenido(){
        echo $this->contenido;
    }
}
?>
```

En este ejemplo se ha creado la clase Caja, indicando como atributos el ancho, alto y largo de la caja, así como el color y el contenido. Se han creado, para empezar, un par de métodos, uno para introducir un elemento en la caja y otro para mostrar el contenido.

Si nos fijamos, los atributos se definen declarando unas variables al principio de la clase. Los métodos se definen declarando funciones dentro de la clase. La variable `$this`, utilizada dentro de los métodos la explicaremos un poco más abajo.

### 10.1.2.- Utilizar la clase

Las clases solamente son definiciones. Si queremos utilizar la clase tenemos que crear un ejemplar de dicha clase, lo que corrientemente se le llama instanciar un objeto de una clase.

```
$micaja = new Caja;
```

Con esto hemos creado, o mejor dicho, instanciado, un objeto de la clase Caja llamado `$micaja`.

```
$micaja->introduce("algo");
$micaja->muestra_contenido();
```

Con estas dos sentencias estamos introduciendo "algo" en la caja y luego estamos mostrando ese contenido en el texto de la página. Nos fijamos que los métodos de un objeto se llaman utilizando el código `"->"`.

```
nombre_del_objeto->nombre_de_metodo()
```

Para acceder a los atributos de una clase también se accede con el código "->". De esta forma:

```
nombre_del_objeto->nombre_del_atributo
```

### 10.1.3.- La variable \$this

Dentro de un método, la variable \$this hace referencia al objeto sobre el que invocamos el método. En la invocación \$micaja->introduce("algo") se está llamando al método introduce sobre el objeto \$micaja. Cuando se está ejecutando ese método, se vuelca el valor que recibe por parámetro en el atributo contenido. En ese caso \$this->contenido hace referencia al atributo contenido del objeto \$micaja, que es sobre el que se invocaba el método.

*Artículo por Miguel Angel Alvarez*

## 10.2.- Constructores en PHP

*Vemos lo que es un constructor y cómo definirlos en programación orientada a objetos en PHP.*

Los constructores son funciones, o métodos, que se encargan de realizar las tareas de inicialización de los objetos al ser instanciados. Es decir, cuando se crean los objetos a partir de las clases, se llama a un constructor que se encarga de inicializar los atributos del objeto y realizar cualquier otra tarea de inicialización que sea necesaria.

No es obligatorio disponer de un constructor, pero resultan muy útiles y su uso es muy habitual. En el ejemplo de la caja, que comentábamos en el anterior [artículo de programación orientada a objetos en PHP](#), lo normal sería inicializar las variables como color o las relacionadas con las dimensiones y, además, indicar que el contenido de la caja está vacío. Si no hay un constructor no se inicializan ninguno de los atributos de los objetos.

El constructor se define dentro de la propia clase, como si fuera otro método. El único detalle es que el constructor debe tener el mismo nombre que la clase. Atentos a PHP, que diferencia entre mayúsculas y minúsculas.

Para la clase Caja definida anteriormente, se podría declarar este constructor:

```
function Caja($alto=1,$ancho=1,$largo=1,$color="negro") {  
    $this->alto=$alto;  
    $this->ancho=$ancho;  
    $this->largo=$largo;  
    $this->color=$color;  
    $this->contenido="";  
}
```

En este constructor recibimos por parámetro todos los atributos que hay que definir en una caja.

Es muy útil definir unos valores por defecto en los parámetros que recibe el constructor, igualando el parámetro a un valor dentro de la declaración de parámetros de la función constructora, pues así, aunque se llame al constructor sin proporcionar parámetros, se inicializará con los valores por defecto que se hayan definido.

Es importante señalar que en los constructores no se tienen por qué recibir todos los valores para inicializar el objeto. Hay algunos valores que pueden inicializarse a vacío o a cualquier otro valor fijo, como en este caso el contenido de la caja, que inicialmente hemos supuesto que estará vacío.

*Artículo por Miguel Angel Alvarez*

## 10.3.- Herencia en PHP

*Hablaremos de esta peculiar característica para hacer copias independientes y personalizadas de clases ya construidas, propia de la programación orientada a objetos.*

La programación orientada a objetos tiene un mecanismo llamado herencia por el que se pueden definir clases a partir de otras clases. Las clases realizadas a partir de otra clase o mejor dicho, que extienden a otra clase, se llaman clases extendidas o clases derivadas.

Las clases extendidas heredan todos los atributos y métodos de la clase base. Además, pueden tener tantos atributos y métodos nuevos como se desee.

Para ampliar el ejemplo que venimos desarrollando, la clase Caja, vamos a crear una clase extendida llamada Caja\_tematica. Esta clase hereda de caja, pero además tiene un "tema", que es la descripción del tipo de cosas que metemos en la caja. Con esto podemos tener varias cajas, cada una con cosas de un tema concreto.

```
class Caja_tematica extends Caja{
    var $tema;

    function define_tema($nuevo_tema){
        $this->tema = $nuevo_tema;
    }
}
```

En esta clase heredamos de Caja, con lo que tenemos a nuestra disposición todos los atributos y métodos de la clase base. Además, se ha definido un nuevo atributo, llamado \$tema, y un método, llamado define\_tema(), que recibe el tema con el que se desea etiquetar la caja.

Podríamos utilizar la clase Caja\_tematica de manera similar a como lo hacíamos con la clase Caja original.

```
$micaja_tematica = new Caja_tematica();
$micaja_tematica->define_tema("Cables y conectores");
$micaja_tematica->introduce("Cable de red");
$micaja_tematica->introduce("Conector RJ45");
$micaja_tematica->muestra_contenido();
```

En este caso, el resultado que se obtiene es parecido al que se obtiene para la clase base. Sin embargo, cuando se muestra el contenido de una caja, lo más interesante sería que se indicara también el tipo de objetos que contiene la caja temática. Para ello, tenemos que redefinir el método muestra\_contenido().

### 10.3.1.- Redefinir métodos en clases extendidas

Redefinir métodos significa volver a codificarlos, es decir, volver a escribir su código para la clase extendida. En este caso, tenemos que redefinir el método muestra\_contenido() para que muestre también el tema de la caja.

Para redefinir un método, lo único que debemos hacer es volverlo a escribir dentro de la clase extendida.

```
function muestra_contenido(){
    echo "Contenido de la caja de <b>" . $this->tema . "</b>: " . $this->contenido;
}
```

En este ejemplo hemos codificado de nuevo el método entero para mostrar los datos completos.

En algunas ocasiones es muy útil apoyarse en la definición de un método de la clase base para realizar las acciones de la clase extendida. Por ejemplo, para este ejemplo, tenemos que definir un constructor para la clase Caja\_tematica, en el que también se inicialice el tema de la caja. Como ya existe un método constructor en la clase base, no merece la pena reescribir el código de éste, lo mejor es llamar al constructor que había definido en la clase Caja original, con lo que se inicializarán todos los datos de la clase base, y luego realizar la inicialización para los atributos de la propia clase extendida.

Para llamar a un método de la clase padre dentro del código de un método que estamos redefiniendo, utilizamos una sintaxis como esta:

```
function Caja_tematica($alto=1,$ancho=1,$largo=1,$color="negro",$tema="Sin clasificación"){
    parent::Caja($alto,$ancho,$largo,$color);
    $this->tema=$tema;
}
```

Aquí vemos la redefinición del constructor, de la clase Caja, para la clase Caja\_tematica. El constructor hace primero una llamada al constructor de la clase base, a través de una referencia a "parent". Luego inicializa el valor del atributo \$tema, que es específico de la Caja\_tematica.



En la misma línea de trabajo, podemos redefinir el método `muestra_contenido()` apoyándonos en el que fue declarado en la clase base. El código quedaría como sigue:

```
function muestra_contenido(){  
    echo "Contenido de la caja de <b>" . $this->tema . "</b>: ";  
    parent::muestra_contenido();  
}
```

Artículo por *Miguel Angel Alvarez*

## Parte 11:

# Epílogos al Manual de PHP

Diversos artículos que finalizan este manual de PHP y ofrecen introducciones a diversos asuntos que son interesantes para conocer sobre este lenguaje de programación. Introducciones a algunas herramientas especialmente útiles, que pueden ayudarnos a desarrollar páginas web.

### 11.1.- Elegir entre PHP4 y PHP5. Conviene la migración?

*A más de dos años de la llegada de la versión 5 de PHP, aún la comunidad de desarrolladores de PHP se plantea el interrogante.*

Las dudas básicamente circulan siempre el mismo camino, y ambas elecciones tienen sus ventajas y desventajas. Intentaremos en este informe orientar a los desarrolladores a decidirse por una u otra alternativa.

Es importante remarcar antes de ubicarse de lleno en el análisis de las ventajas y desventajas de una u otra opción, las principales diferencias existentes entre ambas versiones, cuales son los cambios que repercuten más fuertemente en la compatibilidad de los scripts, y que es lo que nos depara el futuro en toda esta historia.

#### 11.1.1.- Cambios profundos

La llegada de PHP5 vino emparejada de una reestructuración del Core de PHP, lo que los creadores de PHP llama Zend Engine.

Así como el lejano PHP3 incluye su [Zend Engine](http://www.zend.com) 0.5, y PHP4 el Zend Engine 1.0, tenemos Zend Engine 2.0 en PHP5. El cambio de versión no fue trivial; incluye la reescritura casi total del modelo de objetos, entre sus cambios más sustanciales. Esto repercute directamente en los scripts de PHP4 que utilizan clases, tanto en la compatibilidad como en performance de ejecución. Posteriormente en este artículo nos referiremos nuevamente a este tema.

Veamos un ejemplo que nos muestra un cambio sustancial en la implementación del modelo de objetos:

<?

```
class Persona {
    function setNombre($nombre) {
        $this->nombre = $nombre;
    }

    function getNombre() {
        return $this->nombre;
    }
}

function Algo($p) {
    $persona->setNombre("Daniel");
}

1 $persona = new Persona();
2 $persona->setNombre("Pichongol");
3 Algo($persona);
4 echo $persona->getNombre();

?>
```

### 11.1.2.- ¿Cuál es el problema en este código corriendo en PHP4?

En la línea 1 instanciamos un objeto de la clase Persona.

Luego le decimos que se llama Daniel.

El error de implementación viene con la línea 3. El argumento \$p que recibe Algo, no es mas que una copia de \$persona, y eso esta MAL. ¿Porque?, mínimamente por 2 razones.

La primera razón es que esta estrategia es POO-No compatible. Claramente cuando hablamos del Paradigma Orientado a Objetos, estamos casi descartando que cada objeto sea referenciado por su Identificador.

Sin embargo, el Zend Engine 1.0 no está preparado para dicha acción:

```
<?
function ejemplo($val){
    echo $val;
}
$cadena = "texto";
ejemplo($cadena);
?>
```

La variable \$cadena pasada como argumento a la función ejemplo, es copiada para su uso local dentro de dicha función. Es lo que se conoce como paso de parámetros por valor.

El Zend Engine 1.0 hace exactamente esto para todas las funciones, inclusive para las que están dentro de una clase, las cuales en ese caso actúan como métodos:

```
<?
function Algo($persona) {
    $persona->setNombre("Daniel");
}
?>,
```

Volviendo al ejemplo inicial de la clase persona, el método Algo recibe una copia (un clon) del objeto Persona.

La segunda razón viene emparejada con la primera, siendo consecuencia de esta.

Cualquier modificación del objeto Persona que se produzca dentro del método Algo, solo tendrá alcance local, y no se verá reflejado cuando la función retorne.

```
<?
Algo($persona);
echo $persona->getNombre();
?>
```

En ese caso la modificación del nombre que hace la función Algo al objeto Persona no se ve reflejada cuando hacemos echo \$persona->getNombre().

En nuestro browser veremos "Pichongol".

Este es solo un ejemplo del porque de la reestructuración tan importante en el Core de PHP. Es claro que toda

reestructuración barre con cuestiones de compatibilidad, para ganar en otros skills; en este caso claramente estamos ganando en performance, al liberarnos del overhead que implica la constante copia de objetos que son argumentos de métodos y funciones.

En artículos posteriores trataremos en mayor detalle y profundidad los distintos aspectos que fueron modificados, haciendo una comparativa entre como se logran en PHP4 y como se logran en PHP5. Además de explicar profundamente las diferencias en el modelo de objetos nos quedan temas pendientes como Opciones de configuración (php.ini), Conexión a [MySQL](#) (mysqli), cambios en los módulos, etc.

Hecha esta introducción, estamos en condiciones de definir las distintas situaciones en las que se puede encontrar el desarrollador, y que aspectos juegan a su favor o en contra según la situación en la que se encuentre.

### 11.1.3.- ¿Cual es mi escenario?

En el momento de plantearse la pregunta, el desarrollador seguramente se ubicará en alguno de los dos escenarios posibles:  
" Newbie (Iniciación en PHP).  
" Experimentado.

### 11.1.4.- Newbie

En el planteo de esta discusión, podríamos decir que es la situación ideal, o por lo menos la más beneficiosa. Si eres una persona que quiere arrancar en PHP, no lo dudes, PHP5 es para ti. Tus aplicaciones gozaran de las nuevas capacidades en OOP, obtendrás el beneficio de una mejor performance de ejecución (esta comprobado experimentalmente que PHP5 corre un 25% más rápido que PHP4) y tu código estará muy bien acondicionado en cuanto a la compatibilidad con el nuevo hijo que asoma: PHP6.

Por cierto, no todo es color de rosas. Una de los mayores beneficios a la hora de elegir PHP para trabajar en nuestro proyecto es la gran cantidad de código que podemos encontrar en Internet, y utilizarlo para nuestros trabajos. Tenemos una gran probabilidad de que ante alguna tarea que se nos plantea, podamos encontrar algún script que nos solucione la vida, obviamente adaptándolo a nuestras necesidades.

Ahora bien, no todo el código que vamos a encontrar es compatible con PHP5. De hecho la gran mayoría todavía no se ha adaptado. Es cierto que con algún setting en nuestro php.ini podemos ayudar a darle mayor compatibilidad, pero como contrapartida muchas de estas settings se eliminarán en PHP6.

¿Qué queda? Hacerlo compatible modificando el código, una tarea que para un desarrollador que se inicia no siempre es sencillo. De todas formas a no alarmarse, que los grandes proyectos ([PHPNuke](#), [PHPBB](#), etc.) ofrecen compatibilidad.

### 11.1.5.- Experimentado

En este caso, el optar por quedarse con PHP4 o pasar a PHP5 depende de nuestra aplicación.

Las interrogantes que el desarrollador se puede plantear podrían ser:

- ¿Mi aplicación usa clases y objetos?
- ¿Mi motor de Base de datos es MySQL?
- ¿Utilizo un hosting externo?
- ¿Mi aplicación sufre modificaciones en cuanto a los requerimientos y lógica de negocios?

Pasemos a discutir ventajas y desventajas en cada uno de los interrogantes:

### 11.1.6.- ¿Mi aplicación usa clases y objetos?

Como pudimos comprender al comienzo de este artículo, uno de los principales esfuerzos de los diseñadores del Zend Engine radicó en el mejoramiento del modelo de objetos, basándose claramente en un referente indiscutible en esta materia como lo es [Sun](#). Salvando las diferencias, se han tomado muchas cosas de [Java](#), desde convenciones de nomenclaturas hasta estrategias de implementación.

Sería un desperdicio no utilizar dicho esfuerzo, sobre todo si nuestra aplicación hace un uso exhaustivo de clases y objetos.

### 11.1.7.- ¿Mi motor de Base de datos es MySQL?

A diferencia de la estrategia de PHP4 para la conectividad PHP/MySQL, en la que el Core de PHP nos provee de un set de funciones para dicha interacción, en PHP5 MySQL nos provee de un API externo. Básicamente, la razón de este cambio fue una modificación de licencia de MySQL, que obligo a PHP a hacer de MySQL una base de datos más, y no "LA" base de datos, como venia siendo en PHP3 y PHP4. De todas formas, esto no repercute en nuestro código, sino en la performance de nuestra aplicación. El hecho de que una extensión no forme parte del Core de PHP y pase a ser externa nos genera un overhead, una sobrecarga de ejecución en detrimento de la performance. Como contrapartida, PHP5 nos da la posibilidad de sacarle el mayor jugo posible a las muchas mejoras incorporadas en MySQL 4.1.3 o superior, a través del API mysqli. Esto implica hacer uso de otras funciones, modificando nuestro código. Ahora bien, ¿que tan costosa es esta reescritura? Dependerá de nuestra estrategia de conexión a base de datos. ¿Utilizamos una capa de abstracción del estilo [ADOdb](#)? Si la utilizamos estaremos mucho mejor parados frente a tal reescritura. En caso contrario el tiempo invertido será sensiblemente mayor.

### 11.1.8.- ¿Utilizo un hosting externo?

En caso de no disponer de un hosting propio, y tener que depender de un hosting externo que nos provea de PHP, seguramente el hecho de pensar en migrar a PHP5 puede ser un problema. De hecho, estadísticas de principio de 2006 nos indican que solo alrededor del 5% de los hosting que proporcionan PHP, tienen PHP5. Esto no hace mas que reflejar la lentitud con la que se esta moviendo el proceso de traspaso de PHP4 hacia PHP5. Una pregunta que surge directamente sobre este tema es ¿Por qué? Bueno, si uno toma una distribución de Linux, es poco probable que la versión de PHP5 sea la incluida. La conformidad de los programadores con PHP4 es grande, y mucha de la documentación existente esta escrita para PHP4. De todas formas, a no dormirse con PHP4. Un tema que se trata en la segunda parte de este artículo es lo nuevo que nos trae PHP6. Veremos que PHP5 en muchos aspectos es una transición mientras que la confirmación se llama PHP6.

### 11.1.9.- ¿Mi aplicación sufre modificaciones en cuanto a los requerimientos y lógica de negocios?

Cuando las aplicaciones tienen requerimientos de cliente bastante cambiantes, y se emplean recursos para su mantenimiento, o utilizamos una metodología de desarrollo incremental (software versionado), lo ideal es utilizar lo último que nos proporciona nuestra plataforma de programación. Generalmente lo que se busca es un cambio gradual, modular, y sostenido. Por otro lado, si nuestras aplicaciones residen en producción sin mayores modificaciones (algún proceso batch, alguna aplicación depurada, algún algoritmo estable) y estamos conformes con su funcionamiento, quizás no sea de nuestro interés migrar hacia una nueva versión.

Nos queda analizar que hay de nuevo en PHP6 y que cosas deberíamos ir teniendo en cuenta si utilizamos PHP4 o PHP5.

*Artículo por Daniel López*

## 11.2.- Problema del error 404 OK en PHP

*Tenemos que asegurarnos que el error 404, de página no encontrada, sea correctamente enviado al navegador para que no le llegue un HTTP/1.x 404 OK.*

Hay veces que con PHP queremos enviar un error 404 de página no encontrada, para avisar al navegador que una página no existe. Es sencillo enviar por las cabeceras del http un error 404, pero hay que asegurarse que el código de error esté bien enviado para no dar informaciones ambiguas a los clientes que se conecten al servidor.

Esto es importante porque el código 404 también sirve a los buscadores o motores de búsqueda, para informar que una

página no existe. Si damos informaciones ambiguas a los buscadores puede que redunde negativamente en la clasificación de nuestro sitio.

El problema que queremos comentar ahora es el típico error "404 OK". Esto es una ambigüedad: 404 es página no encontrada y OK quiere decir que todo ha ido bien. Pero si no encontró la página ¿Cómo es que decimos que OK? Eso es algo que se puede evitar enviando el error correctamente en la cabecera del HTTP.

Para enviar un código de error 404 por las cabeceras del HTTP con PHP tenemos que hacer lo siguiente:

```
<?
header("HTTP/1.0 404 Not Found");
?>
```

Con la función `header()` de PHP enviamos información en la cabeceras del http. En este caso hemos enviado una cabecera 404 Not Found. Atención, porque la función `header()` debe invocarse antes de haber escrito nada en la página, osea, antes de haber enviado ninguna información al navegador.

Pero dependiendo de la configuración de nuestro servidor esto puede funcionar o no.

La mejor manera de saber qué estamos enviando por las cabeceras del http es utilizar un programa que nos muestre las cabeceras que se generan tanto en el cliente como en el servidor. Existen varios programas que nos podrían servir, pero nosotros vamos a recomendar aquí una extensión de Firefox que hemos comentado ya en otro artículo: Ver [cabeceras HTTP con LiveHttpHeaders](#).

Veremos que en algunas ocasiones enviando esta cabecera el navegador recibe un código de error como este:

HTTP/1.x 404 OK

Eso es algo que tenemos que evitar, porque es ambiguo. La cabecera deseable sería:

HTTP/1.x 404 Not Found

Pues bien, cuando la cabecera que generamos es HTTP/1.x 404 OK tenemos que hacer lo siguiente para conseguir el deseado HTTP/1.x 404 Not Found.

```
<?
header("HTTP/1.0 404 Not Found");
header("Status: 404 Not Found");
?>
```

Primero le indicamos la cabecera del http como 404 y luego lanzamos otra cabecera indicando que el estatus que tiene que enviar es "Not Found". Esto debería funcionar en todos los casos. Es posible que con una sola de las dos cabeceras enviadas nos envíe ya el error 404 Not Found, depende de la configuración de nuestro servidor que necesitemos una u otra. Pero si indicamos las dos en las pruebas que he hecho con varios servidores siempre he recibido el código de error correcto.

Insisto en que lo bueno es que veamos qué cabeceras del HTTP estamos enviando con un producto como [LiveHttpHeaders](#). Así podremos estar seguros de qué cabeceras tenemos que enviar y qué código PHP es el correcto en nuestro caso.

Porque en algunos servidores PHP si enviamos sólo esta cabecera:

```
<?
header("Status: 404 Not Found");
?>
```

Nos envía al navegador el código:

HTTP/1.x 200 OK

Por ejemplo, en este caso es todavía peor, porque nosotros habíamos querido enviar un error 404 y en realidad lo que ha llegado al navegador es un 200 OK que quiere decir página encontrada y correcta.

En definitiva, cabe estudiar detalladamente este asunto para asegurarnos que estamos enviando la cabecera correcta y el error 404 es claro.

Artículo por *Miguel Ángel Álvarez*

## 11.3.- Formulario programado con QuickForm

*Con las clases de QuickForm podrás crear, validar y procesar formularios PHP.*

Todos estaremos de acuerdo en que el poder de php reside en su simplicidad y velocidad.

Una de las aplicaciones más comunes que utilizamos en este lenguaje son los formularios PHP, por su parte PHP no ofrece ninguna función para el desarrollo de los formularios. La biblioteca [PEAR](#), un framework y sistema de distribución de utilidades Php, contiene el paquete [HTML QUICKFORM](#) que proporciona todas las clases y métodos necesarios para manejar formularios HTML.

### 11.3.1.- Paquete de QuickForm

El paquete de [HTML QUICKFORM](#) proporciona un sistema de clases que crean, validan, procesan formularios HTML. En vez de imprimir los elementos del formulario uno a uno, podemos utilizar sus métodos para definir una estructura para el formulario.

QuickForm guarda automáticamente los valores fijados para nuestros elementos a través del envío del formulario, muestra mensajes de error, permite la validación y la filtración que se puede aplicar a los campos individuales y/o al formulario completo y genera el código Javascript para la validación en el lado cliente. Además, simplifica los uploads de archivos. La única razón por la que alguien puede sentirse reticente al uso de QuickForm es la carencia de documentación apropiada. Sin embargo, esto no debe preocuparos pues este artículo va destinado a ello.

Entre otras cosas, nos concentraremos en:

- Mostrar un formulario
- Proceso de entrada
- Validación de campos
- Subidas de archivos
- EL uso SMARTY para optimizar el formulario

La meta de estos artículos es demostrar cómo podemos hacer uso de QuickForm y SMARTY para desarrollar una pequeña aplicación Web.

### 11.3.2.- Requisitos

Hay que destacar que [HTML QUICKFORM](#) no es standard en la instalación de la librería [PEAR](#) por lo que habrá que [descargárselo](#) y colocarlo en el directorio en el que tenemos [PEAR](#).

Es importante saber que es necesario el paquete Observar por favor que el paquete [HTML COMMON](#).

Si la línea siguiente no devuelve un error entonces tienes instalado QuickForm en tu sistema.

```
require ('?HTML/QuickForm.php?');
```

### 11.3.3.- Configurar QuickForm

El primer paso para utilizar el paquete sería incluirlo en nuestro código, deberíamos hacerlo de la siguiente manera:

```
require_once (?HTML/QuickForm.php?) ;
```

El siguiente paso será crear nuestro objeto para el formulario:

```
$form = new HTML_QuickForm(?myform?) ;
```

### 11.3.4.- Elementos del formulario

Podemos añadir nuevos elementos llamando a la función `addElement()`. Podrás definir el orden en el que los elementos se muestran en el formulario.

La función recibe un numero de argumentos, el primero define el tipo de elemento.

Para un formulario estandar los tipos son: text, button, checkbox, hidden, submit, reset, radio, file, image, password, select and textarea.

Ademas [HTML QUICKFORM](#) soporta otros tipos de elementos adicionales como son: date, static, header, html, link, advcheckbox and hiddenselect.

El segundo argumento es el nombre que será usado para el elemento.

El tercer argumento representa la etiqueta de el elemento(el texto que se mostrara en nuestro navegador más tarde).

Algunos de los elementos pueden aceptar argumentos adicionales. Por ejemplo, un elemento "select" puede recibir como argumento un array de valores que son ítems de la lista.

### 11.3.5.- Validación y reglas de filtrado

[HTML QUICKFORM](#) tiene una gran variedad de atributos para su validación, este paquete hace sencillo el uso de estas reglas no solo en el servidor

Artículo por *Manu Gutierrez*

## 11.4.- Librerías JSON para PHP

*Información sobre las distintas librerías y opciones para utilizar la notación de objetos Javascript JSON en programas realizados con el lenguaje PHP.*

JSON es una notación Javascript para escribir objetos que se ha hecho bastante popular en el mundo del desarrollo de webs y que se utiliza en diversos lenguajes de programación, componentes (habitualmente Ajax), etc. Su éxito se debe a que es una excelente forma para almacenar información que deseamos compartir entre distintos componentes o lenguajes de las aplicaciones web. Si trabajamos con Ajax y alguno de los frameworks Javascript existentes, ya habremos notado esta posibilidad y posiblemente ya estemos utilizando JSON o una notación similar.

JSON, cuyas siglas significan JavaScript Object Notation (en español Notación de Objetos de JavaScript), es un formato ligero, fácil de escribir o codificar, así como también es fácil de leer por los seres humanos. Desde Javascript podemos procesar directamente cualquier objeto JSON y existen librerías para la mayoría de los lenguajes de programación que tienen funciones para interpretar este formato. Por ello se ha adoptado universalmente. Para más información podemos visitar el sitio web de JSON en <http://www.json.org/>.

Esto quiere decir que con JSON podemos comunicar datos fácilmente entre scripts Javascript y scripts PHP. Por ejemplo, pensemos en una validación de formulario que se desea hacer con Ajax. Los datos del formulario se pueden enviar a PHP por medio de POST y luego podríamos desde PHP enviar a Javascript el resultado de validar esos datos en el servidor. Como la validación puede ser positiva o negativa, así como puede tener más o menos códigos de error y acciones a realizar dependiendo de la información procesada, el script PHP tiene que mandar una respuesta más o menos elaborada al script Javascript y una posibilidad es enviar esos datos desde PHP utilizando la notación JSON.

### 11.4.1.- Disponibilidad de las funciones JSON en PHP

PHP dispone de varias funciones para hacer distintos tratamientos con notación de objetos JSON, que permite convertir un



objeto PHP, o cualquier otro tipo de variable, a un string con notación JSON, así como crear un objeto PHP a partir de un string codificado con JSON.

En PHP, como decíamos, es posible producir y consumir datos cargados con notación JSON, por medio de unas funciones de las que dispone el lenguaje, que existen de manera predeterminada en los servidores modernos de PHP y que se pueden utilizar también en instalaciones antiguas de PHP, aunque con algún trabajo de instalación adicional.

A partir de PHP 5.2 las [funciones JSON](#) están disponibles siempre, pero si utilizamos por ejemplo PHP 4 tendríamos que instalarlas manualmente. Para ello existen varios paquetes de librerías, que provienen de diversas fuentes, que tienen funciones para tratar con JSON desde PHP.

#### - [Paquete JSON de PECL](#)

Este paquete, del repositorio de librerías PHP PECL, es compatible con PHP 4.3.0 o superior. En versiones superiores a PHP 5.2 está disponible por defecto. En versiones anteriores del lenguaje se tendría que instalar de manera separada.

#### - [Librería JSON-PHP](#)

Es una librería de funciones que se conoce también con el nombre de "Services\_JSON". Originariamente escrita por Michal Migurski, en la actualidad se encuentra disponible dentro del framework PHP PEAR:

[http://pear.php.net/package/Services\\_JSON](http://pear.php.net/package/Services_JSON)

#### - [JSON Lib en Zend Framework](#)

Es una librería de funciones para tratamiento con JSON que forma parte del framework PHP Zend.

#### - [XML-RPC para PHP](#)

Por medio de una extensión para la librería php-xmlrpc, también se puede tratar con notación de objetos JSON.

Las particularidades de cada sistema son ligeramente distintas. Cabe decir que la más interesante sería la primera, que forma parte de PECL, puesto que está escrita en C y por tanto será más rápida de ejecutarse, al formar parte de los componentes nativos de PHP. Como decimos sólo podremos disponer de ella a partir de PHP 5.2, pero en servidores antiguos tendríamos que instalarla aparte, lo que a menudo será complicado, o incluso imposible para sitios web en producción y en un alojamiento compartido.

Así que, si nuestro servidor PHP no dispone de las funciones JSON por estar poco actualizado, nos costará mucho menos esfuerzo y dolores de cabeza utilizar otra librería distinta a la básica de PECL. En este caso, por la facilidad de instalación, se recomendaría la utilización de la librería que viene en PEAR, que se puede descargar del propio sitio de PEAR, en la URL [http://pear.php.net/package/Services\\_JSON](http://pear.php.net/package/Services_JSON)

**Nota:** Hemos publicado en DesarrolloWeb.com en pasados artículos algunas referencias útiles para [instalar PEAR](#) y utilizar algunos [componentes de este framework PHP](#). este framework PHP.

Existe un artículo muy interesante, aunque en inglés, que [compara las diversas librerías existentes para usar JSON desde PHP](#), que contiene diversas informaciones técnicas, requisitos y un análisis del desempeño de cada una.

En el próximo artículo explicaremos cómo [utilizar las funciones JSON en PHP](#) por medio de las funciones nativas del lenguaje.

Artículo por *Miguel Ángel Álvarez*

## 11.5.- CodeIgniter

*CodeIgniter es un framework PHP para la creación rápida de aplicaciones web. Presentación general del framework y primeras notas para empezar a usarlo.*

Probablemente ya sepamos que un [framework](#) es un programa para desarrollar otros programas, CodeIgniter, por tanto, es un programa o aplicación web desarrollada en PHP para la creación de cualquier tipo de aplicación web bajo PHP. Es un producto de código libre, libre de uso para cualquier aplicación.



Como cualquier otro framework, CodeIgniter contiene una serie de librerías que sirven para el desarrollo de aplicaciones web y además propone una manera de desarrollarlas que debemos seguir para obtener provecho de la aplicación. Esto es, marca una manera específica de codificar las páginas web y clasificar sus diferentes scripts, que sirve para que el código esté organizado y sea más fácil de crear y mantener. CodeIgniter implementa el proceso de desarrollo llamado Model View Controller (MVC), que es un estándar de programación de aplicaciones, utilizado tanto para hacer sitios web como programas tradicionales. Este sistema tiene sus características, que veremos en artículos siguientes.

CodeIgniter no es magia, pero contiene muchas ayudas para la creación de aplicaciones PHP avanzadas, que hacen que el proceso de desarrollo más rápido. A la vez, define una arquitectura de desarrollo que hará que programemos de una manera más ordenada y contiene diversas herramientas que ayudan a hacer aplicaciones más versátiles y seguras.

CodeIgniter y otros frameworks PHP pueden ayudarte a dar el salto definitivo como desarrollador PHP, creando aplicaciones web más profesionales y con código más reutilizable, con la diferencia que Code Igniter está creado para que sea fácil de instalar en cualquier servidor y de empezar a usar que cualquier otro framework. Además muchas de sus utilidades y modos de funcionamiento son opcionales, lo que hace que goces de mayor libertad a la hora de desarrollar sitios web.

### 11.5.1.- Características generales de CodeIgniter

Algunos de los puntos más interesantes sobre este framework, sobre todo en comparación con otros productos similares, son los siguientes:

**Versatilidad:** Quizás la característica principal de CodeIgniter, en comparación con otros frameworks PHP. CodeIgniter es capaz de trabajar la mayoría de los entornos o servidores, incluso en sistemas de alojamiento compartido, donde sólo tenemos un acceso por FTP para enviar los archivos al servidor y donde no tenemos acceso a su configuración.

**Compatibilidad:** CodeIgniter, al menos en el momento de escribir este artículo de desarrolloweb.com, es compatible con la versión PHP 4, lo que hace que se pueda utilizar en cualquier servidor, incluso en algunos antiguos. Por supuesto, funciona correctamente también en PHP 5.

**Facilidad de instalación:** No es necesario más que una cuenta de FTP para subir CodeIgniter al servidor y su configuración se realiza con apenas la edición de un archivo, donde debemos escribir cosas como el acceso a la base de datos. Durante la configuración no necesitaremos acceso a herramientas como la línea de comandos, que no suelen estar disponibles en todos los alojamientos.

**Flexibilidad:** CodeIgniter es bastante menos rígido que otros frameworks. Define una manera de trabajar específica, pero en muchos de los casos podemos seguirla o no y sus reglas de codificación muchas veces nos las podemos saltar para trabajar como más a gusto encontremos. Algunos módulos como el uso de plantillas son totalmente opcionales. Esto ayuda muchas veces también a que la curva de aprendizaje sea más sencilla al principio.

**Ligereza:** El núcleo de CodeIgniter es bastante ligero, lo que permite que el servidor no se sobrecargue interpretando o ejecutando grandes porciones de código. La mayoría de los módulos o clases que ofrece se pueden cargar de manera opcional, sólo cuando se van a utilizar realmente.

**Documentación tutorializada:** La documentación de CodeIgniter es fácil de seguir y de asimilar, porque está escrita en modo de tutorial. Esto no facilita mucho la referencia rápida, cuando ya sabemos acerca del framework y queremos consultar sobre una función o un método en concreto, pero para iniciarnos sin duda se agradece mucho.

Sin duda, lo más destacable de CodeIgniter es su accesibilidad, ya que podemos utilizarlo en la mayor gama de entornos. Esta es la razón por la que en DesarrolloWeb.com hemos elegido este framework PHP para comenzar un manual que explicará cómo utilizarlo para desarrollar nuestras propias aplicaciones web. En siguientes artículos iremos contando diferentes aspectos de este framework y lo utilizaremos para crear una primera aplicación web. Para continuar puedes leer el artículo [Instalación y configuración de CodeIgniter](#). También puedes ir al [Manual de CodeIgniter](#) que estamos publicando.

Artículo por Miguel Angel Alvarez