

Manual de PHP



Rubén Alvarez
Miguel Angel Alvarez
Daniel López
Brian Hernández

 desarrolloweb.com

desarrolloweb.com/manuales/manual-php.html

Introducción: Manual de PHP

En este manual de PHP cubrimos todos los aspectos esenciales del lenguaje para comenzar el desarrollo de aplicaciones web del lado del servidor.

Conocerás los principios básicos de la programación en PHP como su sintaxis, estructuras de control, etc. Además explorarás un nutrido conjunto de características y funcionalidades, las necesarias para desarrollar la mayoría de las aplicaciones, como el acceso a la base de datos, el sistema de archivos, etc.

Es un manual de PHP de principio a fin, asequible tanto para programadores como para personas sin experiencia en la programación, que asienta los fundamentos básicos de este popular lenguaje, el más extendido para el desarrollo de aplicaciones web del lado del servidor.

Encuentras este manual online en:

<http://desarrolloweb.com/manuales/manual-php.html>

Autores del manual

Las siguientes personas han participado como autores escribiendo artículos de este manual.

Miguel Angel Alvarez

Miguel es fundador de DesarrolloWeb.com y la plataforma de formación online EscuelaIT. Comenzó en el mundo del desarrollo web en el año 1997, transformando su hobby en su trabajo.



Rubén Alvarez

Rubén es doctor en química y programador aficionado con experiencia en PHP.



Brian Hernández

Desarrollador de apps multiplataforma.

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.

Introducción a la programación en PHP

Explicamos someramente qué es el PHP, sus características principales y los motivos por los que es el lenguaje de programación del lado del servidor más extendido de la web.

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, hasta los más simples y económicos.

La facilidad de PHP se basa en que permite embeber pequeños fragmentos de código dentro de lo que sería una página común creada con HTML. Esos scripts PHP nos permiten realizar determinadas acciones de una forma fácil y eficaz, pudiendo realizar todo tipo de tareas, de las más simples a las más complejas. Esta combinación de PHP dentro del marco de un documento HTML es lo que permite a desarrolladores sin prácticamente nada de experiencia crear comportamientos atractivos de una manera sencilla, una de las claves del éxito del lenguaje. En resumen, con PHP escribimos scripts dentro del código HTML. Como ya estamos familiarizados con HTML, empezar a desarrollar con PHP es prácticamente inmediato. Por otra parte, y es aquí donde reside su mayor interés, PHP ofrece un sinfín de funciones para la explotación de todo tipo de recursos, entre los que destacan las bases de datos, a las que podremos acceder de una manera llana, sin complicaciones.

PHP es lo que se denomina una tecnología del lado del servidor, que ahora se suele englobar dentro del término "Backend". Existen diversos competidores de PHP en el mundo Backend y todos tienen sus cosas buenas y malas. Resultaría muy arriesgado decir que una tecnología o un lenguaje sea mejor o peor que otro, pero sí podemos decir que PHP es el lenguaje preferido por el mayor número de programadores dedicados en el área Backend. Como competidores de PHP podríamos mencionar ASP.NET (o ASP tradicional), NodeJS, Ruby, Java, Python y un largo etc. Sin embargo, en nuestra opinión, si lo que quieres es desarrollar páginas web, el más sencillo y directo con el que podrías empezar es PHP.

Otra de las claves del éxito de PHP es que la mayoría de los CMS más populares (WordPress, Joomla!, Drupal) y los sistemas de comercio electrónico (Prestashop, Woocommerce,

Magento), así como otros cientos de herramientas, están desarrollados en PHP. Por lo tanto, usar PHP es sinónimo de ser capaz de introducirte en muchas herramientas gratuitas y de código abierto para realizar cualquier cosa en el ámbito de la web.

Nota: En otro artículo más detallado podrás encontrar [diversos motivos, y un vídeo, por los que decantarte por aprender PHP](#) como lenguaje para el desarrollo backend.

Algunos aspectos fundamentales de PHP

PHP, aunque multiplataforma, fue concebido inicialmente para entornos Linux y es en este sistema operativo donde se pueden aprovechar mejor sus prestaciones. La mayoría de los servidores de Internet y los hosting soportan PHP sobre sistemas operativos Linux, aunque sin embargo, puedes ejecutar PHP en cualquier otro sistema, obteniendo el mismo soporte y los resultados idénticos. Esto permite que puedas desarrollar PHP en cualquier ordenador, independientemente de si usas Windows, Linux o Mac.

El estilo de programación con PHP es totalmente libre. Puedes usar tanto programación estructurada (funciones) como [Programación Orientada a Objetos](#) (clases y objetos). Incluso algunas características de la programación funcional están siendo incorporadas actualmente. Es por ello que cualquier tipo de programador puede sentirse cómodo con PHP.

PHP presenta una filosofía de código abierto. Existen multitud de herramientas, librerías, frameworks gratuitos que llevan PHP a un nuevo nivel. Además el propio núcleo del lenguaje tiene una de las más nutridas cantidades de funciones para hacer todo tipo de operaciones. No necesitas invertir nada, de dinero, para disponer de un lenguaje poderoso y los mejores complementos para acelerar tu trabajo.

Referencias interesantes para complementar este manual

Este manual de PHP está 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 web. Los capítulos son extremadamente simples, buscando ser accesibles a la mayoría de las personas. Más tarde si lo deseas podrás seguir la lectura de otros manuales dentro de Desarrolloweb.com para ampliar tus conocimientos en distintas áreas.

La forma en la que hemos redactado este manual lo hace accesible a cualquier persona no familiarizada con la programación. Aunque si es tu caso aquí en Desarrolloweb.com tienes un excelente [curso de programación](#) en vídeo, en una serie de clases que impartimos en 2015. Aprenderás todos los conceptos iniciales que debes conocer para afrontar el estudio de cualquier lenguaje, como variables, tipos de datos, estructuras de control, funciones, etc.

Si el lector sabe programar pero tiene poca experiencia, es posible que en determinados momentos pueda verse un poco desorientado. Nuestro consejo en ese caso es no obsesionarse con intentar entender todo antes de pasar al siguiente capítulo. Solo trata de asimilar algunos conceptos y practicar para ir obteniendo soltura. Siempre puedes volver atrás en cuanto una duda surja o cuando 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 también aconsejable, haber leído previamente el manual sobre [manual sobre páginas dinámicas](#), en el cual se explica a grandes rasgos qué es la programación del lado del servidor y por tanto qué es PHP. Esto es interesante porque PHP es un poco particular con respecto a lenguajes tradicionales, ya que para que se ejecute necesitamos un servidor y un cliente que tiene que solicitar una página a ese servidor. Eso es algo sobre lo que incidiremos en varios puntos del manual y en multitud de ejemplos, pero si se tiene claro de antemano el flujo de vida de una página web, desde que se solicita mediante el acceso a una URL desde el navegador, hasta que el servidor la envía al cliente, será mucho mejor.

Más adelante te vendrá bien conocer el [Taller de PHP](#), destinado a analizar de manera práctica la más variada gama de utilidades. 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".

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 26/10/2016
Disponible online en <http://desarrolloweb.com/articulos/12.php>

Breve historia de PHP

Algunos apuntes de la historia de PHP que puedan ser interesantes para el lector que se aproxima a PHP por primera vez.

Como la mayoría del software libre, PHP pertenece a la comunidad. Una gran cantidad de personas ha ayudado a lo largo de su vida a crear tanto el núcleo del lenguaje como la enorme cantidad de librerías que dispone. Sin embargo, debemos atribuir su creación originalmente a Rasmus Lerdorf, creador del lenguaje en 1994.

PHP nació 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. A pesar que por aquel entonces el lenguaje tenía un largo camino por delante para convertirse en una herramienta indispensable, integraba una nutrida cantidad de funcionalidades "de casa", de modo que su comunidad de programadores fue creciendo, atraída por su utilidad y la facilidad para comenzar a desarrollar webs.

PHP en su versión 4 incorporó como novedad el motor "Zend", desarrollado con mayor meditación para cubrir las necesidades de aquel momento 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.

Sin embargo, la madurez definitiva de PHP llegó con la versión 5, que permaneció durante más de 11 años en el mercado y a día de hoy todavía se encuentra en mantenimiento. La principal novedad de la versión 5 fue una mejorada integración del paradigma de la Programación Orientada a Objetos.

Nota: Si bien en la versión de PHP 4 ya se contaba con herramientas para programar con objetos, éstas eran muy rudimentarias y no respondían a las necesidades de los desarrolladores, así como tampoco eran equiparables en potencia y posibilidades a otros lenguajes. PHP, en su intención de servir tanto a programadores experimentados como a desarrolladores que empiezan desde cero, todavía incorpora la posibilidad de desarrollar con o sin programación orientada a objetos.

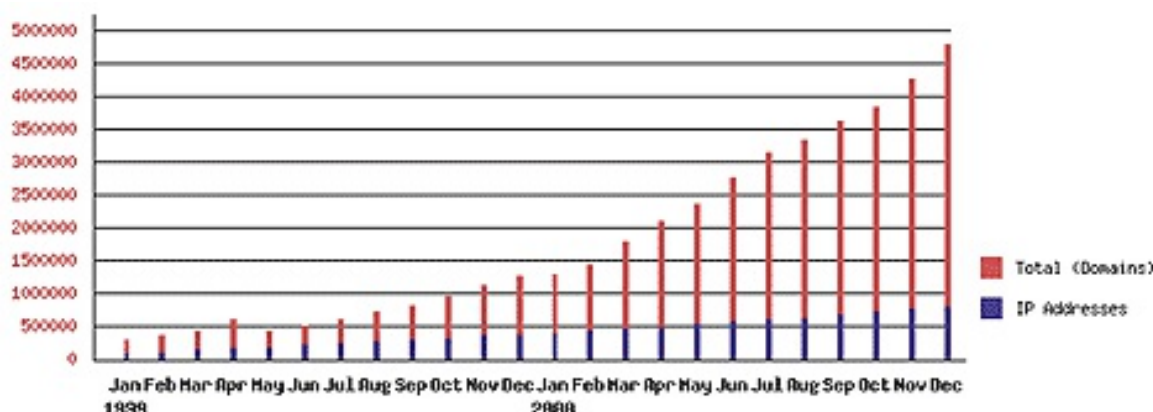
Durante todos los años de vida de PHP 5 hubo muchos cambios. Multitud de herramientas se agregaron al lenguaje, permitiendo hacer cosas que eran altamente demandadas por los desarrolladores y que otros lenguajes más nuevos habían incorporado de salida. Uno de los ejemplos más claros fue el autoloading de clases, lo que permitió la incorporación del [gestor de paquetes Composer](#).

Sin embargo, 11 años con la misma versión sugería que el lenguaje se había estancado y PHP fue perdiendo adeptos, aunque en términos estadísticos, número de desarrolladores y demanda laboral, su superioridad sigue siendo abrumadora. Entre todo ese tiempo varias situaciones hicieron que no se llegara a presentar PHP 6 y finalmente la comunidad decidió saltar ese número de versión y lanzar directamente PHP 7.

PHP 7 a día de hoy es una realidad. Las mejoras en cuanto a rendimiento son muy notables y ha situado de nuevo el lenguaje entre los más poderosos. Está disponible en cantidad de servidores, pero sin embargo su adopción todavía no es total. El motivo es que PHP tiene cantidad de librerías y software que no ha sido totalmente actualizado o que arroja errores al ejecutarse bajo esa nueva versión. En los próximos meses o años la situación cambiará, porque

PHP 7 es muy deseable para cualquier proyecto.

Servidores con PHP



Gráfica del

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

Aunque esta imagen es algo antigua, nos indica que el número de servidores que utilizan PHP se ha disparado, lo que demuestra que PHP es una tecnología muy 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.

A día de hoy, pocas herramientas de entre las más usadas para el desarrollo de sitios o aplicaciones web no están realizadas con PHP. WordPress, Drupal, Magento, Prestashop, etc. son ejemplos de ello. Esa tendencia no para de crecer, pero además ahora le tenemos que sumar una cantidad enorme de frameworks como Symfony, Laravel o Zend, que han permitido convertir a PHP en un lenguaje todavía más poderoso, productivo y capaz de implementar las mejores prácticas para la salud de los proyectos.

Comunidad de PHP

También la comunidad de desarrolladores que usa PHP ha evolucionado mucho. Muchos de los profesionales que vienen usando este lenguaje a lo largo de los años han crecido, profesional y naturalmente, al lado de PHP. Podemos decir que PHP ha sido y sigue siendo el causante de su éxito o sustento profesional.

Esa madurez de los desarrolladores también ha sido importante para el lenguaje. En sus inicios la comunidad era atraída a PHP por su cantidad de utilidades y la facilidad con la que comenzar a trabajar. Sin embargo, las personas no se preocupaban tanto con aspectos como la seguridad o la mantenibilidad de las aplicaciones. Hoy la comunidad es consciente de la importancia de plataformas robustas y escalables y ello ha permitido que PHP haya dado un vuelco profesional. Mucha de esa transformación se la debemos a los mencionados frameworks y a la capacidad de PHP de absorber y traer para sí lo mejor de otros lenguajes.

En el Manual de PHP aprenderás a dar tus primeros pasos con el lenguaje, pero queremos que no te quedes ahí y sigas esforzándote para aprender más y más. En DesarrolloWeb.com tienes cantidad de material para seguir creciendo, como el [manual de la programación orientada a objetos de PHP 5](#), el [Manual de Composer](#) o de [frameworks como Laravel](#). Ser riguroso con tu trabajo y cómo usas el lenguaje es el mejor favor que harás a la comunidad y a ti mismo como profesional.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 26/10/2016
Disponible online en <http://desarrolloweb.com/articulos/12.php>

Tareas principales del lenguaje PHP

Mencionamos los principales grupos de funciones integradas en el lenguaje PHP, a nivel general, y lo que ellas nos ofrecen.

PHP 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.

En este artículo vamos a exponer una pequeñísima muestra de las cosas que PHP te ofrece para el desarrollo de aplicaciones web, pero no pienses que son las únicas. Simplemente las hemos seleccionado por ser las tareas que resultan más habituales en cualquier tipo de proyectos y que han sido resueltas en PHP tradicionalmente de una manera sencilla y al alcance de cualquier persona, incluso sin demasiados conocimientos de programación.

Además, todas las funcionalidades recogidas en el presente artículo permiten hacerse una idea del tipo de cosas que aprenderás en el [Manual de PHP básico](#).

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.

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.

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.

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.

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.

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 01/01/2001
Disponible online en <http://desarrolloweb.com/articulos/12.php>

Cómo instalar PHP y MySQL

En los siguientes capítulos vamos a explicar cómo crear tu entorno de trabajo para empezar a trabajar con PHP. Explicaremos diversos modos que existen para instalar PHP y la base de datos MySQL, complemento habitual en las aplicaciones web con PHP. Comenzaremos con un repaso general a todas las alternativas posibles para crear ese entorno de trabajo, con distintos niveles de complejidad y adaptados a distintos perfiles de desarrolladores y necesidades de equipos de trabajo. Luego iremos a la parte más práctica, que consiste en instalar programas que nos facilitan la instalación, con un único paso, de todas las herramientas necesarias para ejecutar PHP en un ordenador, sin necesidad de entrar en ninguna configuración en la mayoría de los casos.

Guía definitiva para crear tu entorno de desarrollo para PHP

Guía completa para la creación de tu entorno de desarrollo con PHP para sistemas Windows, Linux y Mac, tanto para personas que se inician como para desarrolladores experimentados.

Ha llovido mucho desde que redactamos nuestro [Manual de PHP básico](#). A lo largo de todos estos años ha cambiado mucho nuestra manera de trabajar con este lenguaje y también las herramientas disponibles para ser más productivos y trabajar en entornos más profesionales.

En este artículo pretendemos no solo actualizar nuestra guía de PHP, sino también ofrecer una vista de pájaro de todas las opciones disponibles en la actualidad para instalar PHP y MySQL más un servidor web donde poder ejecutar los sitios que vayamos desarrollando. Además queremos clasificar las distintas maneras según su dificultad, de modo que analizaremos tanto los entornos más básicos y sencillos de usar, como los más avanzados y profesionales.

En este artículo no pretendo dar una guía paso por paso para instalar PHP, algo que ya se ha explicado en otros artículos ya publicados en DesarrolloWeb.com, sino aclarar las distintas alternativas de creación de tu entorno de desarrollo, básico, intermedio y avanzado. De este modo el interesado podrá tener una buena idea de cómo puede crear su entorno de desarrollo para comenzar y cuáles serían los siguientes pasos y posibilidades si quiere incrementar las prestaciones, en un entorno más productivo o profesional. Cada una de las posibilidades de instalación se acompañará de enlaces a referencias con guías específicas para conseguir una rápida y sencilla configuración.



Qué programas necesitas instalar para trabajar con PHP

Antes que nada, para ayudar a los usuarios más inexpertos, conviene aclarar que programas necesitas para poder empezar a desarrollar con PHP. Son pocos:

- PHP, el propio lenguaje.
- Una base de datos, que suele ser MySQL pero que podría ser cualquier otra. Es importante porque cualquier proyecto básico suele apoyarse para su desarrollo en un sistema gestor de base de datos donde almacenar la información.
- Un servidor web, que puede ser comúnmente Apache o Nginx. Es importante porque para que se pueda servir una web, el navegador tiene que acceder a un servidor. Además, PHP lo más normal es que se instale como módulo de un servidor web, de modo que las páginas generadas por el servidor se puedan procesar mediante PHP.

Como servidor web la alternativa más común con diferencia es Apache, aunque se podría ejecutar PHP sobre otra serie de servidores web. El propio Apache es multiplataforma, igual que PHP, y lo tenemos disponible en Linux, Mac o Windows. Además de ser la solución más habitual, Apache es también la que tiene más opciones avanzadas. Aunque hoy la alternativa Nginx nos ofrece un servidor ligero y rápido, preferido por muchas personas. En entornos Windows también podrías instalar PHP sobre un IIS, aunque esa alternativa es menos común.

No obstante, no debes marearte por tantas posibilidades en cuanto a servidores, ya que el funcionamiento de PHP es independiente del servidor donde se esté ejecutando. Tengas uno u otro, el procesado y resultado de ejecución de PHP será el mismo.

Qué tipo de entornos de desarrollo podemos usar

Una vez aclarados los distintos programas que necesitas y antes de entrar en el detalle sobre cómo instalar PHP quiero describir brevemente las posibilidades de instalación de PHP. En siguientes puntos de este artículo entraremos en detalle con cada punto, pero antes queremos que se conozcan todos.

Nota: Ahora observarás que te indico varias alternativas. No quiero con ello despistar y parecer que comenzar con PHP es difícil, sino ofrecer un poco de cultura general y clarificar cuáles serían los siguientes pasos si ahondas en el mundo de PHP. Si todo esto te parece "hablar en chino", no te preocupes, quédate con la opción "instaladores todo-en-uno" y verás que tienes PHP funcionando en tu ordenador en 5 o 10 minutos.

Instaladores todo-en-uno

Lo más cómodo, rápido y directo es instalar todos los programas necesarios de una única vez, a través de uno de los muchos paquetes de instalación que luego mencionaremos. Esta alternativa es la más recomendada para la mayoría de las personas que empiezan y son muy útiles porque permiten contar, en un único paso, con todo lo que necesitas para trabajar. No solo te instalará todos los programas mencionados antes, sino que además los configurarán correctamente para trabajar entre ellos.

Con estos instaladores todo en uno podrás en minutos tener PHP, Apache y MySQL y comenzar a desarrollar sin complicaciones. Si estás empezando con PHP y quieres aprender a desarrollar es la opción más interesante.

Instalación de todos los programas por separado

Solo para los usuarios de Linux mi recomendación sería instalar todo lo que necesitas por separado, por medio de los correspondientes repositorios. Es muy sencillo y si trabajas en Linux seguro sabes de lo que estoy hablando. Existen Todo-en-uno para Linux pero no es la manera natural de trabajar en este sistema.

Virtualización

Para los usuarios más avanzados existe la posibilidad de virtualizar. Básicamente consiste en instalar en tu ordenador una máquina virtual y ejecutar tus aplicaciones desarrolladas con PHP en esa máquina virtualizada (guest) y no en tu sistema real (host).

La virtualización es muy útil porque nos permite trabajar en un entorno de desarrollo que será lo más parecido al entorno de producción (aquel donde vas a poner tu aplicación web en funcionamiento). Es el entorno más profesional y nos ahorra determinadas complicaciones habituales que enumeramos a continuación.

1. Aunque PHP funciona igual en cualquier sistema cuando estás desarrollando, puedes tener diversos módulos instalados que igual no se encuentran en el entorno de producción, o viceversa. Eso puede hacer que tus páginas funcionen en un sitio y no en otro.
2. Para equipos de trabajo, donde cada uno desarrolla con un sistema operativo y con programas de diversos tipos, trabajar con una virtualización permitiría a todo el equipo ejecute la aplicación en el mismo entorno virtualizado, con las mismas librerías, servidores, mismas versiones, etc. Eso ahorra la posibilidad de que una aplicación no funcione a un desarrollador y sí a otro.
3. Finalmente, al trabajar con un entorno virtualizado te obligas a trabajar con un dominio real, no accederás a tu proyecto con localhost y cada proyecto podrá tener sus propias configuraciones, sin que unos interfieran en otros. Al pasar de un proyecto a otro simplemente apagarás una máquina virtual y encenderás otra, serán independientes y no corres el riesgo que, al cambiar configuraciones para un proyecto, deje de funcionar el otro.

La virtualización la considero muy útil cuando se desarrollan aplicaciones realmente complejas, donde tienes gran número de dependencias, pero agrega dificultad para empezar. Es muy importante cuando se trabaja en equipo, sobre todo a medida que crece el número de desarrolladores. Pero no la recomendaría para una persona que está empezando con PHP, aunque sí me parece importante que se conozca que existe esa posibilidad.

Editores online

No quiero perder la oportunidad de nombrar otra alternativa interesante en la actualidad, si lo que quieres es probar PHP o trabajar desde cualquier ordenador y no depender de llevarte de un lugar a otro tu código y la instalación de servidores.

Los editores online permiten comenzar a usar PHP sin tener que instalar nada y programar sin salirte del propio navegador. Están muy bien, porque permiten aprovechar todas las ventajas de la nube y además, como en la virtualización, trabajar con entornos más reales, similares a los que usarías en producción

Los editores online tienen habitualmente entornos de desarrollo ya listos para trabajar y no necesitas instalar nada, simplemente hacer login en el editor online y comenzar a usarlo. Pero a decir verdad pueden ser un poco sofisticados para quienes están comenzando, ya que la mayoría están enfocados a desarrolladores con algo de experiencia. Si eres como yo, que toda la ofimática te la has llevado a la nube, apreciarás la posibilidad de trabajar con editores online, ya que podrás del mismo modo trabajar remotamente en cualquier ordenador, o incluso en el tablet, sobre el mismo proyecto. Aunque, a decir verdad, aunque lo he intentado, no he llegado nunca a usarlos para ningún proyecto.

Referencias para instalación de tu entorno de trabajo con PHP

Ahora que ya te deben quedar claras las posibilidades, quiero poner algunas referencias para que puedas saber cómo comenzar realmente en tu tarea de poner a punto tu ordenador para trabajar con PHP.

Instaladores todo en uno para Windows

Para Windows yo recomiendo Xampp, que me parece el más completo y también el más usado, por lo que si tienes cualquier necesidad de configuración avanzada es más probable que encuentres documentación y ayudas de otras personas que hayan tenido esa misma necesidad o problema. Encuentra [más información de Xampp](#). Aunque si no te funciona por cualquier motivo hay otras alternativas como [Wamp](#). No son los únicos del mercado, pero creo que sí los más importantes y recomendables.

Instaladores todo en uno para Mac

En el caso de Mac el que es más usado con diferencia es Mamp, que resulta muy interesante. La versión gratuita es muy sencilla, pero tiene lo que necesitas para comenzar. La versión de pago es muy avanzada y tiene unas prestaciones muy elevadas y configuraciones realmente útiles que puedes hacer a golpe de ratón. Si de verdad te dedicas profesionalmente a PHP creo

que te interesa tener la versión profesional. Aprende [más sobre Mamp](#).

Instalar en PHP en Linux

Como decía antes, si trabajas en Linux te recomiendo instalar PHP, aunque sea para comenzar, por medio de los repositorios de tu distro. Es muy sencillo y tienes una guía paso a paso aquí para [instalar PHP en Ubuntu \(o distros basadas en Debian como el propio Debian o Mint\)](#).

Entornos por medio de virtualización

Si deseas explorar esta posibilidad te recomendamos [comenzar con Vagrant](#), que es la alternativa más sencilla de crear entornos de desarrollo que puedes compartir fácilmente con el resto del equipo, para que todos trabajen sobre una virtualización idéntica. Es gratuita y además existen diversas herramientas relacionadas que nos permiten crear virtualizaciones por medio de asistentes muy sencillos de usar.

Editores online

Existen varios pero creo que una buena alternativa para comenzar a explorarlos es Cloud9. Puedes encontrar [más información de cloud9 aquí](#).

Conclusión

Espero que con esta serie de posibilidades te hayamos aclarado el camino sobre cuáles son las opciones para trabajar con PHP. Espero que tantas alternativas no hayan servido para despistar, sino para cubrir un amplio espectro sobre el que puedas decidir.

A continuación en el Manual de PHP encontrarás más información detallada sobre cada una de estas alternativas principales. Nos centraremos en aquellas que nos permiten instalar todos los programas cómodamente con una única instalación, y en Linux la alternativa de instalar los softwares por separado, ya que es más recomendable.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *31/10/2016*
Disponible online en <http://desarrolloweb.com/articulos/guia-crear-entorno-desarrollo-php.html>

Instalar PHP fácilmente en Windows

Cómo instalar PHP en sencillos pasos en un ordenador con Windows. En 5 minutos tendrás una instalación rápida y sencilla de PHP con Xampp o WampServer.

En este artículo vamos a abordar un par de programas para la instalación sencilla de PHP en un sistema operativo Windows, con la que puedes crear tu entorno de desarrollo fácilmente y

empezar a desarrollar PHP en pocos minutos.

Usaremos programas de instalación automática, que nos permiten contar con PHP, MySQL y el servidor web Apache, en un solo paso y además configurados correctamente para que no tengamos que preocuparnos por nada, solo comenzar a desarrollar. En el mercado existen varias alternativas para realizar esto y nosotros comentaremos un par de ellas, que son las más tradicionales para Windows: Xampp y Wamp. Existen otras, que van apareciendo (y también desapareciendo), por lo que preferimos contaros las que a nosotros nos han funcionado bien siempre y son proyectos con bastante buena salud hasta la fecha.

Ten en cuenta solo que estas instalaciones son indicadas para tu ordenador personal, aquel donde vas a desarrollar con PHP. Para el servidor donde pondrás tu aplicación PHP en funcionamiento no serían indicadas. Además que, aunque PHP funcione perfectamente sobre Windows, los servidores de PHP en producción suelen ejecutarse sobre el sistema Linux.

Instalar PHP en Windows Con Xampp

Antes de Xampp, u otras herramientas similares, instalar PHP en Windows era una tarea medianamente compleja, que requería varios pasos. Hay que instalar el servidor web Apache, luego el propio PHP, configurarlos para trabajar juntos, etc. Adicionalmente, tendrás que instalar un motor de base de datos como MySQL o cualquier otro sistema gestor que prefieras usar. Sin embargo, si tu objetivo es disponer de PHP en el ordenador donde vas a desarrollar aplicaciones web, es mucho más recomendable usar un instalador rápido.

Nuestro preferido y el que te recomendamos en principio es Xampp, aunque no es el único y cada desarrollador puede tener una opinión distinta. Puedes obtener este paquete desde su propia página web: <https://www.apachefriends.org/es/index.html>

Estamos seguros que cualquier persona que sepa instalar un programa en Windows sabrá también instalar PHP con Xampp, pues es un sencillo programa con un asistente que tienes que seguir paso por paso, como cualquier otro instalador. Te ofrece pocas opciones de configuración y nuestro consejo es que al principio las aceptes todas las que te vienen de manera predeterminada (al menos para personas que están empezando).

Nota: La única parte que quizás cambiaría de Xampp es cuando te dice los módulos que quieres instalar. Si sabes que algo no vas a necesitar puedes simplemente quitarlo, para ahorrar espacio en disco, pero por lo demás, siempre conviene ir a las opciones por defecto. Lo que no cambiaría es la ruta de la carpeta donde estarán los archivos de la web, tu document root. Si alguna vez quieres personalizar cosas de tu Xampp y buscas tutoriales más avanzados en Internet, generalmente te guiarán dando por hecho que el document root es el que se ha marcado por defecto.

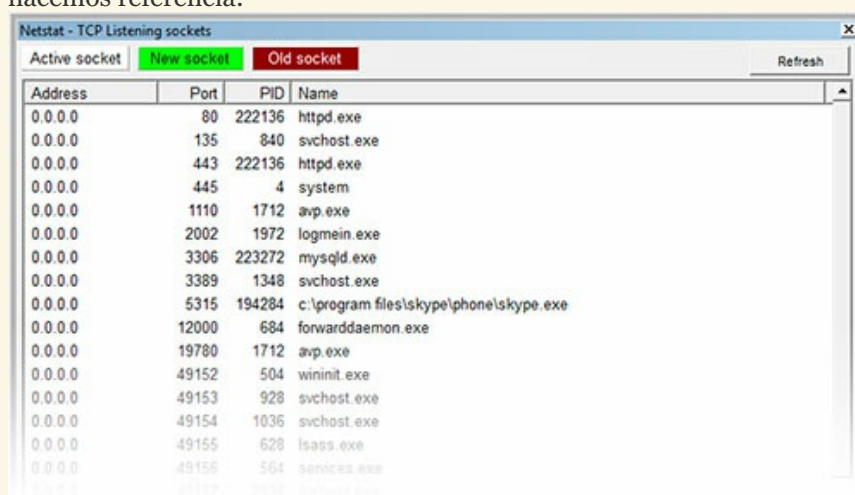
Para explicar otros detalles de Xampp te recomendamos la lectura del [artículo de descripción de Xampp](#), que es antiguo pero bastante actualizado, ya que no han cambiado prácticamente nada.

En DesarrolloWeb.com encontrarás también otros artículos dedicados a cosas más avanzadas

a realizar con Xampp, como es el caso de crear dominios personalizados para los sitios que tengas en desarrollo, lo que te permite entrar mediante un dominio de Internet (como a cualquier otro sitio web) en lugar de desde localhost. Eso es bueno porque así podrás ejecutar tus aplicaciones con un entorno más real y parecido a como estarán publicadas una vez las pongas en producción. Si te interesa saber más, consulta el artículo [Configurar virtualhost en Apache para Windows](#). Además tenemos un [videotutorial de Xampp](#) que también te puede ser de utilidad

Nota: A modo de advertencia, ya que es un error muy recurrente y aunque ya lo hemos tratado en otras ocasiones en DesarrolloWeb... Apache usa el puerto 80 para funcionar. Si ese puerto está ocupado por otro programa, ya sea Skype o IIS o cualquier otro programa, no se podrá arrancar. La solución sería cambiar el puerto donde Apache funciona o mejor, cambiar la configuración o detener ese otro programa que cause interferencias. El propio Xampp cuando lo ejecutas tiene una sencilla herramienta para detectar los puertos abiertos donde puedes ver qué programa es el que tiene ocupado el puerto 80, si es que has encontrado este problema en tu equipo.

La siguiente imagen te muestra la herramienta de gestión de puertos de Xampp a la que hacemos referencia:



The screenshot shows a window titled "Netstat - TCP Listening sockets". It has three tabs: "Active socket", "New socket", and "Old socket". The "Active socket" tab is selected. Below the tabs is a table with columns: Address, Port, PID, and Name. The table lists several listening sockets on 0.0.0.0, including ports 80, 135, 443, 445, 1110, 2002, 3306, 3389, 5315, 12000, 19780, 49152, 49153, 49154, 49155, and 49156, each associated with a specific process like httpd.exe, svchost.exe, system, avp.exe, logmein.exe, mysqld.exe, forwarddaemon.exe, wininit.exe, lsass.exe, and services.exe.

Address	Port	PID	Name
0.0.0.0	80	222136	httpd.exe
0.0.0.0	135	840	svchost.exe
0.0.0.0	443	222136	httpd.exe
0.0.0.0	445	4	system
0.0.0.0	1110	1712	avp.exe
0.0.0.0	2002	1972	logmein.exe
0.0.0.0	3306	223272	mysqld.exe
0.0.0.0	3389	1348	svchost.exe
0.0.0.0	5315	194284	c:\program files\skype\phone\skype.exe
0.0.0.0	12000	684	forwarddaemon.exe
0.0.0.0	19780	1712	avp.exe
0.0.0.0	49152	504	wininit.exe
0.0.0.0	49153	928	svchost.exe
0.0.0.0	49154	1036	svchost.exe
0.0.0.0	49155	628	lsass.exe
0.0.0.0	49156	564	services.exe

Instalar PHP fácilmente en Windows con Wamp

Para los que no han instalado nunca PHP hay que comentar que no tengan miedo para nada al lenguaje, que es muy sencillo y agradecido. No cabe tampoco tener miedo de la instalación o configuración de PHP, porque en este artículo vamos a explicar un modo de realizarla en 5 minutos y sin necesidad de conocimientos iniciales.

Os paso unas notas rápidas sobre el proceso de instalación de PHP en Windows por medio de un programa que se llama Wamp Server 2, que os facilitará la tarea. Podrás comenzar a trabajar con PHP en 5 minutos!!

Nota: Volvemos a insistir que nuestro preferido es Xampp, que está mucho más actualizado. No encuentro un motivo para instalar Wamp que no sea que el propio Xampp te esté dando problemas en tu sistema, en cuyo caso puedes probar con Wamp como alternativa.

Lo primero es entrar en la página de Wamp Server, que es la siguiente:

<http://www.wampserver.com/en/>

Como había comentado, Wamp Server es un programa que instala en un sólo paso Apache + PHP + MySQL y los configura para trabajar juntos.

Habría que hacer la descarga de la última versión de Wamp Server en:

<http://www.wampserver.com/en/download.php>

En la página de descarga te especifica claramente la lista de programas que va a instalar, así como las versiones de los mismos. En el momento de escribir este artículo iban por la versión WampServer 2.0, que instala esta lista de programas:

- Apache 2.2.8
- PHP + PECL
- SQLitemanager
- MySQL 5.0.51b
- Phpmyadmin

Nota: La lista de programas o versiones de los lenguajes que te ofrece Wamp puede variar durante el tiempo a criterio de los mantenedores del software.

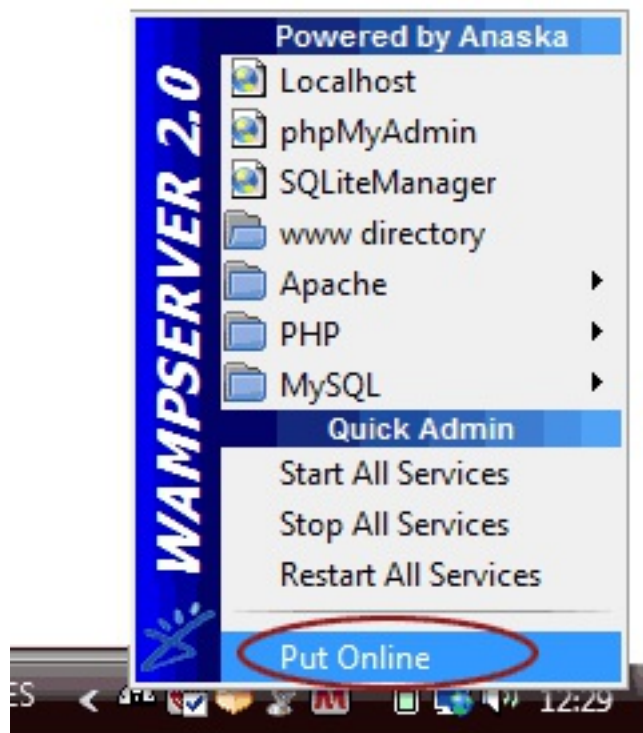
Una vez descargado el programa, lo ejecutamos para realizar la instalación de Wamp Server 2. La instalación se basa en un asistente normal que nos solicitará varios datos típicos de instalaciones, como que aceptemos los términos de la licencia. Luego nos saldrá la ventana para acabar que marcaremos que ejecute Wamp Server inmediatamente.



Si todo ha funcionado, en 1 minuto más podremos comprobar si PHP 5 está funcionando en nuestro ordenador. Sólo tendríamos que encender los servicios. Para ello Wamp Server tiene un panel de control que se accede desde un icono de programa residente de la barra de tareas. Tiene una forma rara, como un cuentakilómetros. Lo veremos en esta imagen:



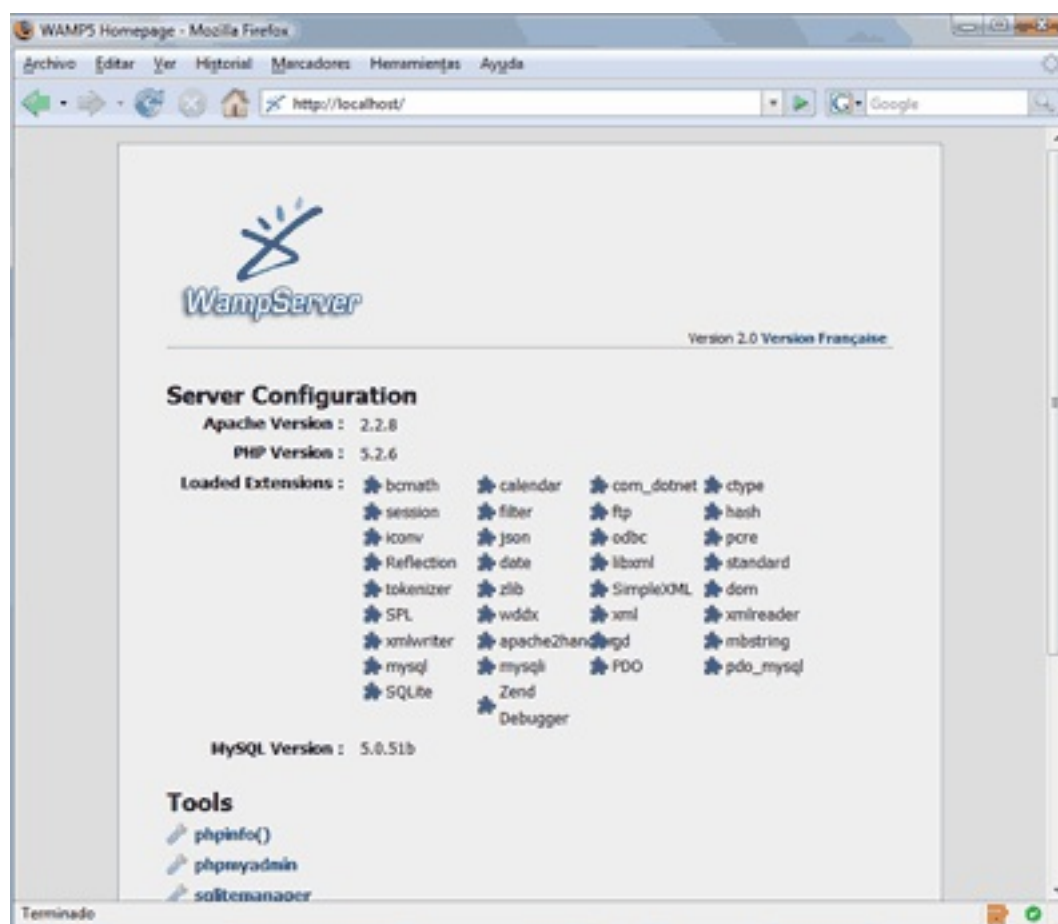
Pulsamos el icono con el ratón (botón izquierdo, clic normal) y veremos abajo del todo una instrucción que pone "Put Online", que pondrá todos los servicios en funcionamiento.



Ahora, para comprobar que los servicios funcionan sólo nos queda abrir un navegador. Vamos a escribir la siguiente dirección URL en la barra de direcciones:

<http://localhost>

Entonces nos tiene que salir la página de inicio del servidor Apache con PHP 5, personalizada por Wamp, que es algo como esto:



Si no sale nada puede que haya habido un problema o un error al iniciar los servicios, generalmente el Apache, que utiliza el puerto 80 que a veces está ocupado por otro programa como Skype o IIS. Lee la FAQ: [No funciona el Wamp Server 2](#).

Ahora podremos colocar en nuestro servidor todas las páginas PHP que queramos probar o los proyectos que hayamos creado anteriormente. El directorio donde generalmente se localiza la raíz de publicación es: C:/wamp/www

En esa carpeta podríamos subir cualquier archivo PHP 4 o PHP 5 y debería ejecutarse perfectamente. Otra cosa que puede fallar es que los inicios de bloques de código PHP que debéis utilizar son con “<?php” y no sólo con “<?”, que está deshabilitado por defecto.

Os aseguro que leer este artículo os llevará más tiempo que instalar PHP 5 en vuestro ordenador. Con Wamp Server 2 es muy fácil.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 31/10/2016
Disponible online en <http://desarrolloweb.com/articulos/instalar-php-5.html>

Instalar Lamp, Apache - MySQL - PHP, en Linux

Cómo instalar todos los programas que necesitas para comenzar a desarrollar

con PHP en entornos Linux: Apache, PHP y MySQL, lo que se conoce como Lamp. Para Distribuciones basadas en Debian como Ubuntu o Linux Mint.

En DesarrolloWeb.com hemos explicado en diversos artículos cómo [instalar todos los programas necesarios para empezar a trabajar con PHP en local sobre sistemas Windows](#), pues tradicionalmente hemos orientado nuestros tutoriales a los usuarios de dicho sistema. Sin embargo, no debemos de olvidarnos de los usuarios de cualquiera de los otros sistemas operativos y en este artículo le toca a GNU/Linux.

De hecho, si se me permite la apreciación, aunque PHP es multiplataforma, su entorno más natural para ejecución es Linux, ya que la mayoría de los servidores PHP corren bajo ese sistema operativo. Por ello, para los que somos desarrolladores y que además nos interesa aprender un poco de administración de servidores, no nos vendría nada mal tener nuestro PHP ejecutando bajo una instalación de Linux. Además, se trata de algo realmente sencillo.

Nota: Existen otros tutoriales diversos en DesarrolloWeb.com que pueden interesarte si lo que quieres es [instalar PHP en otros sistemas operativos](#).

Debe haber decenas de maneras de instalar Apache, PHP y MySQL sobre GNU/Linux y en Internet encontraremos una gran cantidad de información a este respecto. Nosotros vamos a destacar una a continuación que nos ha funcionado siempre bien, con algunos detalles adicionales que nos pueden facilitar diversas labores de desarrollo en nuestros sistemas. Además, complementaremos la información comentando los pasos para instalar otra herramienta fundamental, como es el PhpMyAdmin.

Instalar los paquetes por línea de comandos con apt-get

Vamos a instalar todos los paquetes necesarios para poder trabajar con PHP y lo haremos sobre línea de comandos, que es un método que difícilmente podrá fallarnos. Para ello utilizaremos el comando apt-get de sistemas basados en Debian.

Nota: Yo estoy trabajando sobre Ubuntu, por ello voy a utilizar un comando que viene de sistemas Debian para la descarga de paquetes que es el apt-get. Ese comando, como decía, está disponible en las distros Debian, pero también en aquellas basadas en Debian, como Ubuntu o Linux Mint. Ubuntu es el tercer sistema operativo más utilizado, creado por Canonical, y Mint sería el cuarto en importancia a nivel mundial. Si utilizas otro sistema GNU/Linux que no esté basado en Debian, esta información quizás no te servirá de mucho.

Conviene decir que esta receta la hemos obtenido del [sitio web HowtoForge](#). Yo la he traducido libremente y la he complementado con explicaciones adicionales y algunos pasos extra que he considerado de interés.

Como un primer paso, podemos lanzar un comando para obtener los privilegios de root para la administración a través de la línea de comandos.

```
sudo su
```

Nos pedirá la clave de nuestro usuario, que es la misma clave que usamos al arrancar el equipo.

Paso 1: Instalación de MySQL

Ahora instalaremos MySQL, para lo que ejecutamos el siguiente comando:

```
apt-get install mysql-server mysql-client
```

Durante el proceso de instalación tendrás que introducir la clave que deseas configurar al usuario root de MySQL.

Paso 2: Instalar Apache

Ahora toca instalar el servidor web Apache, en su versión 2. Para instalar Apache2 lanzamos el siguiente comando:

```
apt-get install apache2
```

En este punto, una vez instalado Apache, puedes hacer una primera comprobación, que nos permitirá saber si el servidor web está funcionando correctamente. Para ello abre un navegador cualquiera e introduce la URL de localhost o la IP local de tu ordenador:

```
http://localhost
```

O bien:

```
http://192.168.0.55 (sustituye esa IP por la IP de tu máquina)
```

Nota: Si no sabes cuál es tu IP de red local, tienes a tu disposición en Linux el comando `ifconfig`.

Al acceder a cualquiera de esas dos URL debería salirte el mensaje de Apache diciendo que está funcionando (It works!).

Para tu información, el directorio predeterminado donde se guardan los documentos del servidor web en Apache es `/var/www` y el fichero de configuración del servidor está en `/etc/apache2/apache2.conf`. Otras configuraciones están almacenadas en otros subdirectorios

de `/etc/apache2` como `/etc/apache2/mods-enabled` para los módulos habilitados, `/etc/apache2/sites-enabled` para los "virtual hosts" y `/etc/apache2/conf.d` para las configuraciones globales que afectarán a todos los virtual host.

Nota: Existe en DesarrolloWeb.com un manual sobre la instalación de Apache en Windows, que no te ayudará mucho si estás en Linux, pero que tiene mucha [información sobre la configuración de Apache](#). Te servirá de ayuda puesto que la mayoría de los archivos de configuración funcionan exactamente igual y las variables de configuración son exactamente las mismas.

Paso 3: Instalar PHP

El siguiente paso es Instalar PHP. La versión actual en el momento de escribir este artículo es la 5, que se instalaría con el siguiente comando.

```
apt-get install php5 libapache2-mod-php5
```

Después de la instalación de PHP5 como módulo de Apache, debemos reiniciar el servidor web y para ello lanzamos este otro comando.

```
/etc/init.d/apache2 restart
```

Como puedes ver, hacemos un "restart", pero también podrías hacer primero un "stop" y luego un "start".

Llegado este punto, podemos crear ya un documento web que nos sirva para comprobar si PHP está correctamente instalado en nuestro sistema y para ello podemos utilizar la línea de comandos y el editor que prefieras. A algunas personas les gusta el editor llamado Vi, pero yo soy de los que prefiere otros más sencillos como el Joe.

Nota: Para conocer más acerca del programa Joe, puedes visitar la FAQ: [¿Sabéis de una alternativa a Vi?](#)

En cualquier caso, tendrás que crear un archivo en la ruta por defecto de publicación del Apache, que ya dijimos es `/var/www`. Puedes llamarle como desees, por ejemplo `info.php`, en el que colocarás las siguientes líneas de código.

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

Esa función de PHP `phpinfo()` te mostrará mucha información útil sobre la instalación de PHP que tienes en tu máquina, como módulos incluidos y diferentes configuraciones en funcionamiento.

Nota: para crear el archivo PHP también puedes utilizar el editor gráfico que desees, que te será incluso más cómodo que cualquier programa que trabaje con el terminal. El único detalle es aplicarle los permisos necesarios a la carpeta `/var/www` para que sea propiedad de tu usuario y puedas ciertas cosas con ella. Ten en cuenta que ponerse como usuario dueño de la carpeta se recomienda solo sitios que no estén en producción. Lo harías con el comando:

```
sudo chown -R $USER:$USER /var/www
```

Si lo deseas, también puedes hacer el comando :

```
echo $USER
```

Para ver el nombre del usuario en el que estás y el que vas a poner como dueño de la carpeta `/var/www`.

Una vez creado el archivo de texto `info.php` con el código indicado, podemos acceder a él desde un navegador con una URL como esta:

```
http://localhost/info.php
```

También puedes sustituir "localhost" por tu dirección IP de red local.



Si ves toda una serie de información de tu instalación PHP, como en la imagen anterior, es que has podido instalar PHP en correctas condiciones.

Paso 4: Instalar módulo PHP5-mysql y otras extensiones PHP necesarias

Si haces scroll hacia abajo en la página del phpinfo() podrás ver el listado de módulos PHP que tienes disponibles. Puede que no tengas todos los que necesitas y en concreto podrás observar que no tienes habilitado el módulo de MySQL, por lo que podrás instalarlo.

Si haces el comando:

```
apt-cache search php5
```

Podrás encontrar el listado de extensiones de PHP disponibles en los repositorios, puedes elegir aquellos que desees e instalarlos con la línea:

```
apt-get install php5-mysql php5-curl php5-gd php-pear php5-imagick php5-sqlite php5-tidy php5-xmllrpc php5-xsl
```

A continuación debes reiniciar el servidor para que los cambios tengan efecto.

```
/etc/init.d/apache2 restart
```


Si refrescas la página del `phpinfo()` podrás encontrar los nuevos módulos instalados.

Paso 5 opcional: Instalar PhpMyAdmin

Seguramente agradecerás contar con una copia de PhpMyAdmin para gestionar tus bases de datos MySQL. Para ello puedes invocar el comando:

```
apt-get install phpmyadmin
```

Verás a continuación una serie de preguntas, como tu servidor web. Una vez instalado puedes acceder al gestor por medio de una URL como esta:

```
http://localhost/phpmyadmin/
```

Conclusión

Con este proceso tendrás a tu disposición todos los materiales para comenzar a desarrollar con PHP en tu Linux.

Simplemente señalar un detalle importante, que ya se apuntó de refilón, pero que quiero remarcar. Es posible que en esta instalación de Apache y PHP quieras desarrollar sitios web y utilizar para programar dichos sitios un editor para programadores con interfaz gráfica como Komodo Edit o Eclipse. En este caso recuerda que, para editar o crear archivos en la carpeta `/var/www` desde esos programas, tendrás que poner tu usuario como dueño de la carpeta, con `chown`. Eso se explicó en una nota anterior.

Referencia: Para optimizar la configuración de tu Apache en Linux de una manera muy práctica para tu servidor de desarrollo, te aconsejamos la lectura del artículo [Configuración de Apache en Linux con carpetas externas](#).

De momento es todo! Espero que te sirva!

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 19/04/2012
Disponible online en <http://desarrolloweb.com/articulos/instalar-php-apache-mysql-linux.html>

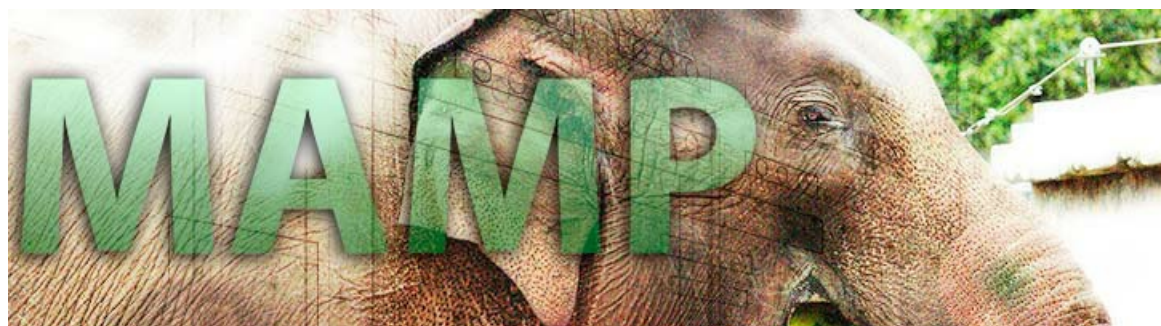
Instalar PHP en Mac con Mamp

Instalar PHP con Apache y MySQL en un ordenador Mac OS X usando el paquete todo en uno Mamp. Primeros pasos y configuraciones de Mamp.

Uno de los temas más recurrentes dentro de DesarrolloWeb.com es la instalación de PHP. Tenemos decenas de artículos y FAQs creadas a lo largo de los ya casi 15 años de historia de este sitio web. Sin embargo, nunca habíamos abordado la instalación de PHP sobre Mac OS X, tutorializada y paso a paso como nos gusta siempre ofrecerte la información.

Ahora pues, vamos con una de las pendencias para completar los primeros pasos para los que se introducen en PHP que ya te ofrecemos a través del [Manual de PHP Básico](#). Para simplificar las cosas y haceros la vida más fácil a todos, abordaremos este asunto a través de uno de esos instaladores todo-en-uno que nos ayudan a tener en pocos instantes instalados y configurados todos los paquetes de software necesarios para poder ejecutar PHP en nuestro ordenador.

En Mac se usa tradicionalmente el sistema llamado Mamp, que es el que os vamos a enseñar en este artículo, sin embargo, tampoco es el único. Lo cierto es que todos los "maqueros" que conozco usan Mamp, pero la verdad es que si se desea también se puede encontrar versiones para OS X de paquetes populares como Xampp.



Realmente, para instalar el Mamp poco te tememos que decir. Si ya eres usuario de Mac desde hace tiempo no encontrarás problema alguno, ya que es el proceso que has usado en decenas de programas que tendrás instalados en tu máquina. Por ello y para darle un poco más de utilidad a este texto, te explicaremos también cuáles son los primeros pasos y las configuraciones básicas que puedes querer hacer cuando empieces a usar Mamp. Al final de este texto encontrarás también un vídeo que he grabado para explicar estas mismas cosas pero de manera visual.

Mamp o Mamp Pro

Como decimos siempre, para trabajar con PHP lo más cómodo es tener en local un servidor web, donde crearás tus páginas durante la etapa de desarrollo. Luego las páginas las subirás a un servidor web de Internet para que estén disponibles para todo el mundo y a cualquier hora. Por tanto, para trabajar con PHP necesitarás de tres elementos básicos. Apache que es un servidor web, el módulo de PHP (para que Apache sea capaz de interpretar y ejecutar páginas con PHP), y MySQL (una base de datos con la que construir tus aplicaciones).

Estos tres programas los tienes en Mamp para instalar todo en uno y de manera sencilla. Además te los entregan configurados perfectamente para que puedas comenzar a usarlos en el instante. Tanto la versión "Mamp" como su hermano mayor "Mamp Pro" contienen estos tres ingredientes básicos, por tanto, cualquiera de las dos será suficiente. Mamp (a secas) es gratuito, por lo que representa la mejor opción para comenzar. ¿Entonces qué más consigo si tengo Mamp Pro?

Realmente pocas personas que conozco usan Mamp Pro, pero nuestro compañero de la comunidad de DesarrolloWeb.com y profesor de EscuelaIT @micromante usa el Mamp Pro, porque tiene alguna cosilla que a él le viene bien profesionalmente. El precio es más que razonable y tiene varias mejoras interesantes como configuración del servidor de email para envío de correo desde páginas PHP, posibilidad de elegir entre muchas versiones de PHP, acceso al servidor a través de tu red local, etc. Pero la que más me ha llamado la atención por su utilidad y porque es algo que a veces hacer a mano te da algún que otro problemilla es la posibilidad de usar lo que se llaman "virtual host".

Nota: Si eres nuevo en PHP seguramente no te diga nada eso del "virtual host". En ese caso no te preocupes, pero para los que quieran saber a qué me refiero es crear una especie de servidor independiente para cada uno de los sitios que quieras alojar (para cada proyecto de cada cliente). De ese modo puedes acceder al proyecto con un nombre de dominio algo como micliente.example.com, lo que te ofrece un entorno bastante más parecido a como tendrás publicado el sitio una vez esté en el servidor remoto y dominio definitivo. Además, al ser host independientes tienes la oportunidad de configurar cosas también de manera independiente, sin que afecte a otros proyectos que tengas en tu mismo ordenador.

Primeros pasos con Mamp

La versión básica de Mamp es muy sencilla, pero hay unas pocas cosas que puedes saber para facilitarte un poco más su uso.

Encender y apagar los servicios:

Desde la ventana de administración de Mamp puedes encender y apagar los servicios (Apache y MySQL). Hay un sencillo botón para esta tarea que no tendrás problemas en localizar desde la ventana principal. Solo ten en cuenta que el servidor web solo estará disponible (así como todas las páginas que cuelgues de él) cuando el servidor esté encendido. Un problema típico de no poder acceder a una página en tu servidor es que te hayas olvidado de iniciar los servicios.

Puerto:

El Mamp configura de manera predeterminada Apache para escuchar en un puerto determinado, por defecto el 8888. Es por ello que la ruta de tu servidor en local es:

<http://localhost:8888>

La palabra "localhost" es un alias de tu ordenador local y luego con ":8888" indicas el puerto donde tu servidor web está configurado. Este puerto lo puedes cambiar también desde la interfaz de administración del Mamp.

Nota: generalmente los servidores web trabajan con el puerto 80. En esos casos no es necesario indicar el puerto, porque tu navegador lo usará de manera predeterminada para acceder a los sitios web. Solo en el caso que el servidor web esté configurado en un puerto diferente, es necesario indicarlo en la URL. Osea, <http://localhost> es lo mismo que escribir

<http://localhost:80>. Mamp configura Apache en otro puerto para que no tengas incompatibilidades con otras aplicaciones que puedan estar usando también ese mismo puerto.

Directorio de publicación:

Otra de las cosas que debes aprender es a localizar es el directorio "raíz" de publicación de tu servidor web. Es muy fácil de localizar en la instalación básica de Mamp. Simplemente vas a "Preferencias / Apache" y lo encontrarás. De manera predeterminada está en tu disco duro, directorio "aplicaciones/MAMP/htdocs". Si lo deseas puedes cambiarlo, para situarlo en otra carpeta diferente, pero no te recomiendo hacerlo a no ser que ya tengas un poquito de experiencia.

Lo importante del directorio de publicación es que es el lugar donde vas a colocar todos los archivos que quieres que estén disponibles a través de tu servidor web. En esa carpeta podrás meter tanto páginas HTML como PHP, archivos CSS, imágenes, JS, etc. Osea, todo lo que haya en tu proyecto o en cada uno de los proyectos que tendrás en el servidor. Para acceder a estos archivos basta con escribir la ruta de tu servidor <http://localhost:8888> y luego la ruta para acceder a ese archivo desde el directorio raíz de publicación.

<http://localhost:8888/directorio/archivo.php>

Creo que con estos conceptos iniciales tienes suficiente para comenzar sin temor a liarte más de la cuenta. Recuerda que en DesarrolloWeb.com tienes muchas otras ayudas para profundizar en PHP en la sección [PHP a Fondo](#).

Vídeo de instalación de Mamp

En el siguiente vídeo puedes ver como un novato como yo en los sistemas OS X instala Mamp en 3 minutos. Además te explicaré las cosas más importantes que debes saber para no liarte en tus primeros pasos usando tu servidor web local.

Para ver este vídeo es necesario visitar el artículo original en:

<http://desarrolloweb.com/articulos/instalar-php-mac-mamp.html>

Este artículo es obra de *Miguel Angel Alvarez*

Fue publicado por primera vez en 18/08/2014

Disponible online en <http://desarrolloweb.com/articulos/instalar-php-mac-mamp.html>

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.

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.

Después de varios capítulos del [Manual de PHP](#) en los que hemos introducido el lenguaje, sus características y cómo instalar nuestro entorno de trabajo, estamos seguros que tendrás muchas ganas de comenzar a ver código. Así que vamos con ello!

En este capítulo vamos a explicar la sintaxis básica y cómo en una página HTML podemos mezclar el código del lenguaje de marcación (HTML) con el código del lado del servidor (PHP). Verás que es bien sencillo, motivo por el cual a los desarrolladores que ya saben HTML les resulta muy sencillo comenzar con PHP. Además veremos algunas cosas básicas y consejos interesantes para que tu código PHP se pueda ejecutar perfectamente en cualquier tipo de servidor.

Apertura y cierre del código PHP

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 del código escritas en este lenguaje. Esto se hace, como en otros casos, delimitando nuestro código por etiquetas de apertura y cierre. 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, algo de lo que hablaremos en seguida.

Estos son los modos de abrir y cerrar las etiquetas que delimitan el código PHP:

```
<? y ?>  
<?php y ?>
```

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 (Generalmente .php, pero podría configurarse el servidor para que busque

código PHP en otras extensiones de archivo...) para ejecutar los bloques de scripts PHP.

El servidor, antes de enviar la página al navegador se encargará 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.

Nota: En PHP 7 se eliminaron otros estilos de apertura y cierre del código PHP: `<% y %>`, así como `<script language="php">`.

En PHP la apertura del código con el tag en su versión corta (`<?>`) no se encuentra siempre activado por defecto. Es algo que depende del servidor y de la versión de PHP que esté instalada en él. Para evitar problemas debidos a la plataforma donde se ejecuta PHP no te recomendamos utilizarlo. No obstante, si tienes la oportunidad de alterar la configuración del lenguaje PHP (mediante la edición del archivo `php.ini` correspondiente, del que hablaremos en otro momento), podrías definir que también se interprete esa etiqueta mediante la directiva "[short-open-tags](#)".

Uso de ; para delimitar sentencias

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 ultima expresión, la que va antes del cierre de etiqueta, este formalismo no es necesario.

```
<?php echo 'código PHP' ?>
```

Aunque la sentencia "echo" anterior (que sirve para escribir desde PHP salida en la propia página) no acaba en ";" el código es perfectamente válido, porque inmediatamente después tenemos el cierre del script PHP.

Comentarios en PHP

Incluimos también en este capítulo la sintaxis de comentarios, que funcionan muy similares a los de otros lenguajes como Java, C o Javascript.

Nota: 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 su ejecución, sino a nosotros mismos u otros desarrolladores que puedan leer ese código más adelante. 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:

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

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.

Ejemplo completo de una página PHP

Ahora veamos un código completo de lo que podría ser una página PHP. Verás que comenzamos con una página básica escrita con HTML en la que hemos insertado un código PHP. El código de momento es lo de menos, lo importante es ver cómo se integra el código PHP en una página HTML.

```
<!doctype html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Primera página PHP</title>
</head>
<body>
  <h1>Esto es HTML</h1>
  <?php
  echo '<p>Esto viene de PHP</p>';
  ?>
</body>
</html>
```

Para poder probar esta página PHP deberías nombrarla con extensión ".php". Podría ser algo como "pag1.php" o "index.php". Luego tendrás que colocarla en el directorio de publicación de tu servidor ("document root" en inglés), cuyo depende de cuál sea el servidor que estés usando para poder comenzar con PHP. Normalmente esa carpeta se llama algo como "htdocs",

"httpd", "htdocs", "www", etc. Luego, teniendo el servidor Apache (o el servidor que tengas en tu caso) encendido, tendrás que acceder a la página a través de "<http://localhost/pag1.php>". Aunque esto depende mucho de cómo tengas configurado tu entorno de trabajo. En los artículos de instalación de PHP se ofrecen más detalles sobre cómo puedes configurar el entorno de trabajo y cuáles serían los directorios y modos de acceso a tus archivos PHP desde cada tipo de servidor.

Nota: Otra cosa que nos gustaría mencionar, aunque pueda resultar un tanto avanzada, es que en la práctica es interesante seguir una serie de buenas prácticas, como la separación del código por responsabilidades o el uso de sistemas de templates. Quizás es demasiado pronto para mencionarlo, pero lo cierto es que el hecho de PHP permitirnos mezclar el código HTML con el código PHP puede producir a la larga proyectos con un difícil mantenimiento. Ahora no es el momento de preocuparte por ello, si es que estás empezando con PHP, pero es bueno que lo tengas en cuenta para más adelante y que consultes, aquí en DesarrolloWeb.com artículos y manuales más avanzados donde te explicamos cómo codificar de modo que te asegures que tu proyecto será ordenado y lleno de buenas prácticas. Siempre está bien tener una mirada crítica en nuestro trabajo a fin de explorar aquellas prácticas y herramientas que nos permitan ser mejores profesionales.

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 29/10/2016
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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 asunto dentro del [Manual de PHP](#), explicando también los tipos de datos que podremos encontrar en el lenguaje.

Anteriormente en DesarrolloWeb.com, en el manual de páginas dinámicas, ya habíamos introducido el [concepto de variable](#). No obstante podemos entender una variable como un dato almacenado en una referencia. Técnicamente una variable apunta a una posición de la memoria, donde se almacena un dato. Las variables se utilizan en los lenguajes de programación para darle un nombre a ese dato, a esa posición de la memoria, de manera que se pueda entender o saber lo que contiene. Al final, esos datos almacenados son los que se utilizan para conseguir los resultados de los programas.

Por su parte, un tipo de datos es la característica de un dato almacenado. Es decir, si es de una forma concreta, numérica, alfanumérica, etc. Todos los lenguajes tipan de alguna manera los datos, aunque algunos son más permisivos que otros a la hora de realizar operaciones con variables de distintos tipos.

Nota: Si tienes cualquier duda sobre lo que son las variables o los tipos de datos te recomendamos asistir a la grabación de esta excelente clase donde te lo explican de una manera detallada. [Variables y tipos en los lenguajes de programación](#).

\$ en el nombre de las variables

En el capítulo anterior ya comentábamos que, para PHP, las variables eran definidas comenzando siempre por el símbolo dólar (\$). Es quizás una de las características más fuertes del lenguaje. Al ver el dólar al principio del nombre de cualquier variable podrás rápidamente deducir que tal código está escrito en PHP.

Las variables siempre deberían tener un nombre descriptivo sobre lo que ellas van a almacenar. Por tanto, al nombre de una variable en PHP le colocaremos el símbolo \$.

```
<?php $total = 300 ?>
```

Tipos de datos en PHP

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

Variables numéricas

Este tipo de variables almacena cifras, números, que pueden tener dos clasificaciones distintas:

- **Enteros** \$entero=2002; Numeros sin decimales
- **Reales** \$real=3.14159; Numeros con o sin decimal

Variables alfanuméricas

Este tipo de datos almacena textos compuestos, cadenas de caracteres, que pueden contener letras, símbolos y números o cifras.

- **Cadenas** Almacenan variables alfanuméricas \$cadena="Hola amigo";

Boleanas

Este tipo de variables almacena un valor lógico, que puede valer verdadero o falso. Es muy común en la programación este tipo de variables boleanas.

- **Boleano verdadero** \$verdadero = true;
- **Boleano falso** \$falso = false;

Matrices, tablas o arrays

Es un tipo de datos en el que, en lugar de tener un dato, podemos almacenar un conjunto de ellos, a los que accedemos a través de índices. Cada una de las casillas de un array o los datos de nuestra matriz a su vez almacenan informaciones numéricas y/o alfanuméricas, pudiendo

mezclar tipos de variables de manera arbitraria entre sus distintas casillas.

- Arrays Son las variables que guardan las tablas \$sentido[1]="ver"; \$sentido[2]="tocar"; \$sentido[3]="oir"; \$sentido[4]="gusto"; \$sentido[5]="oler";

Más adelante encontrarás capítulos enteros dedicados a los [arrays en PHP](#).

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. Entraremos con detalle en los objetos más adelante, ya que su complejidad hace difícil explicarlas ahora.

PHP tiene tipado dinámico

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 (PHP hará todo lo posible por interpretar nuestra operación, aunque técnicamente no tenga mucho sentido hacer determinadas operaciones):

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

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.

Nota: Los lenguajes como PHP que permiten mayor flexibilidad en los tipos de las variables se dicen que tienen **tipado dinámico**. En ellos una variable puede tener distintos tipos a lo largo de su vida, es decir, a medida que el programa se ejecuta una variable podrá cambiar de tipo. Generalmente durante el procesamiento del programa se va infiriendo los tipos de las variables, en tiempo de ejecución, según el tipo de datos del valor que se le asigna o las operaciones que se realizan sobre ellas. Otra manera de referirse a este tipo de lenguajes de programación es "débilmente tipados", aunque esta segunda denominación es menos correcta, porque puede inducir a una comprensión errónea, ya que

en la realidad las variables siempre tienen tipos, aunque estos puedan variar con el tiempo.

PHP es sensible a las mayúsculas y minúsculas

PHP entiende de manera distinta las mayúsculas y minúsculas. En el caso del nombre que le damos a una variable, no es lo mismo escribirla con mayúscula o minúscula, o mezclando mayúsculas y minúsculas de distinta manera. Por tanto, hay que tener mucho cuidado a la hora de escribir los nombres de variables, y no cambiar mayúsculas por minúsculas, ya que PHP entenderá dos variables distintas aunque nosotros podamos intentar referirnos a la misma. Cuando estamos empezando quizás sea un buen consejo trabajar asignando nombres a las variables siempre en minúsculas, para evitar este tipo de malentendidos a veces muy difíciles de localizar.

En el caso que tengamos una variable con un nombre compuesto de varias palabras, en PHP es una práctica común colocar la variable toda en minúscula y separar las palabras por guiones bajos.

```
<?php $mi_variable_bonita = "me gusta PHP" ?>
```

Variables asignadas por referencia

En PHP también podemos asignar variables por referencia, aunque a decir verdad no es una característica que se use mucho. 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](#)

Este artículo es obra de *Rubén Alvarez*

Fue publicado por primera vez en 30/10/2016

Disponible online en <http://desarrolloweb.com/articulos/12.php>

Cambio del tipo de las variables en PHP

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

En el artículo anterior del [Manual de PHP](#) comenzamos a explicar [cómo se trabaja con variables](#) en este lenguaje. Ya comentamos que PHP tiene un tipado dinámico, pero hay algunas cosas que puedes necesitar hacer en el día a día y que debes saber para cubrir el conocimiento fundamental de variables y tipos de datos en PHP.

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. Esto es básicamente lo que se llama "tipado dinámico" o "tipado débil", característica no sólo de PHP, sino de muchos otros lenguajes como Javascript.

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

```
$cadena = 'esto es una cadena';  
$cadena = 34 //La variable $cadena cambió de 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?
```

En este artículo veremos dos tipos posibles de alteración del tipo de variables, más allá del propio que hace PHP con el comportamiento derivado de su tipado dinámico. A esta operación se la conoce habitualmente como "Forzado".

Forzado

Variar el tipo de datos que contiene una variable con el tiempo es una cosa que no siempre es aconsejable, porque si no tenemos certeza de si una variable contiene un dato de un tipo u otro, a veces los resultados obtenidos pueden no ser los esperados.

Para evitar problemas en muchas ocasiones puede venir bien realizar el forzado de una variable a un tipo en concreto, de manera explícita, lo que nos permitirá saber que cuando llega el flujo del programa a un punto dado, aquella variable tendrá el tipo de datos esperado. En PHP existen diversas maneras de forzar una variable a un tipo.

Establecer el tipo con settype()

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"

Casting de variables

Hay otra manera de realizar un forzado, para que una variable se comporte como un tipo determinado. Ahora vamos a ver otro mecanismo de forzado que es similar al de otros lenguajes como C o Java.

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

Los forzados permitidos son:

- (int), (integer) - fuerza a entero (integer)
- (real), (double), (float) - fuerza a número con decimales (coma flotante)
- (string) - fuerza a cadena (string)
- (array) - fuerza a array (array)
- (object) - fuerza a objeto (object)
- (unset) - fuerza a null
- (binary) - fuerza a "binary string"

Conclusión

Si estás comenzando con PHP y la programación en general, quizás este tema del cambio de tipo y el forzado puede parecer una información un tanto avanzada o sin una aplicación clara. Si es así no te preocupes demasiado por ahora, pero ten presente que tú como programador eres capaz de cambiar los tipos de las variables, para que tus programas hagan exactamente lo que tú deseas.

Cuando hay una incongruencia de tipos PHP siempre intenta hacer lo más adecuado con el código que ejecuta, pero no siempre la solución que él toma es la que tú pudieras pensar. En

esos casos, el forzado será realmente importante. Sin duda cuando tengas más experiencia con el lenguaje estas situaciones irán apareciendo.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 01/12/2016
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 25/04/2006
Disponible online en <http://desarrolloweb.com/articulos/12.php>

Variables de sistema en PHP

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

En los anteriores artículos del [Manual de PHP](#) hemos abordado el tema de las variables. Hasta ahora hemos explicado cómo crear nuestras propias variables y almacenar valores, pero si hablamos de variables en PHP no podemos dejar de mencionar a las variables de sistema.

Para entender las variables de sistema tienes que apreciar que PHP es un lenguaje que se ejecuta en el servidor, bajo demanda de un cliente. Por tanto, la ejecución de PHP se produce dentro de un marco muy concreto, donde intervienen varios actores, principalmente el cliente (generalmente el usuario que entra usando su navegador) y el servidor (donde se ejecuta el código PHP, que básicamente debe producir la salida que se enviará al cliente).

Ahora que has asimilado la naturaleza de PHP como lenguaje de lado servidor, debes de entender que en ese marco existen diversas informaciones que pueden ser útiles a la hora de ejecutar aplicaciones web. Dentro de una página PHP tendremos por tanto acceso a toda una serie de variables que nos informan sobre nuestro servidor y sobre el cliente que ha solicitado una determinada página. A estas informaciones, que podemos recoger en forma de variables, les llamamos "variables de sistema".

Nota: 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.

`$_SERVER`

La mayoría de las variables de sistema las podemos recibir a partir de un array denominado `$_SERVER`.

Nota: `$_SERVER` es un array asociativo, cuyos índices son cadenas de texto y no números. Aún no hemos abordado el tema de [arrays en PHP](#), pero lo veremos con detalla más adelante.

Técnicamente `$_SERVER` se conoce como una "variable superglobal", de la que hablaremos en este mismo artículo un poco después. Existen multitud de datos asociados al array `$_SERVER`, algunos sin utilidad aparente y otros realmente interesantes y con una aplicación directa para

nuestras aplicaciones web. Aquí os enumeramos algunas de estas variables y la información que nos aportan:

- `$_SERVER["HTTP_USER_AGENT"]` Nos informa principalmente sobre el sistema operativo y tipo y versión de navegador utilizado por el internauta. Su principal utilidad radica en que, a partir de esta información, podemos redireccionar nuestros usuarios hacia páginas optimizadas para su navegador o realizar cualquier otro tipo de acción en el contexto de un navegador determinado.
- `$_SERVER["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.
- `$_SERVER["HTTP_REFERER"]` Nos indica la URL desde la cual el internauta ha tenido acceso a la página. Muy interesante para generar botones de "Atrás" dinámicos o para crear nuestros propios sistemas de estadísticas de visitas.
- `$_SERVER["PHP_SELF"]` Nos devuelve una cadena con la URL del script que está siendo ejecutado. Muy interesante para crear botones para recargar la página.
- `$_SERVER["HTTP_GET_VARS"]` Se trata de un array que almacena los nombres y contenidos de las variables enviadas al script por URL o por formularios GET.
- `$_SERVER["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.
- `$_SERVER["HTTP_COOKIE_VARS"]` Se trata de un array que almacena los nombres y contenidos de las cookies. Veremos qué son más adelante.
- `$_SERVER["PHP_AUTH_USER"]` Almacena la variable usuario cuando se efectúa la entrada a páginas de acceso restringido. Combinado con `$_SERVER["PHP_AUTH_PW"]` resulta ideal para controlar el acceso a las páginas internas del sitio.
- `$_SERVER["PHP_AUTH_PW"]` Almacena la variable password cuando se efectúa la entrada a páginas de acceso restringido. Combinado con `$_SERVER["PHP_AUTH_USER"]` resulta ideal para controlar el acceso a las páginas internas del sitio.
- `$_SERVER["REMOTE_ADDR"]` Muestra la dirección IP del visitante.
- `$_SERVER["DOCUMENT_ROOT"]` Nos devuelve el path físico en el que se encuentra alojada la página en el servidor.
- `$_SERVER["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 tendrá un valor definido, a menos que el internauta acceda al script a partir de un enlace desde otra página.

Si quieres ver cuál es el conjunto completo de las variables del sistema que dispones dentro de `$_SERVER` en tu entorno, es suficiente con escribir y ejecutar una página PHP que contenga este código:

```
<?php
var_dump($_SERVER);
?>
```

Eso realizará un listado de todo el contenido del array asociativo `$_SERVER` y lo mostrará como salida en la página web.

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 y a las que se puede acceder desde cualquier punto del código PHP.

Nota: Estas variables ya existían anteriormente en PHP, aunque se accedían desde otros arrays. Si lees artículos antiguos de PHP, o ya conoces PHP desde hace mucho tiempo, te puede aclarar que algunas de estas variables superglobales se accedían antes por medio de los arrays del tipo `$HTTP_*_VARS`. Por ejemplo `$_GET` antes era `$HTTP_GET_VARS` o `$_POST` era antes `$HTTP_POST_VARS`. La forma antigua de referencia a las variables superglobales todavía se puede activar en algunos servidores, desde el `php.ini` con la directiva `register_long_arrays`.

La lista de estas variables superglobales de PHP es la siguiente:

`$GLOBALS`

Contiene una referencia a cada variable disponible en el espectro de las variables del script. Las llaves de esta matriz (índices del array) 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. Es equivalente a lo que antes se conocía como `$HTTP_SERVER_VARS`. Son las variables de sistema que hemos explicado antes en este artículo.

`$_GET`

Variables proporcionadas al script por medio de HTTP GET. Es equivalente a lo que antes se conocía como `$HTTP_GET_VARS`.

`$_POST`

Variables proporcionadas al script por medio de HTTP POST. Es equivalente a lo que antes se conocía como `$HTTP_POST_VARS`.

`$_COOKIE`

Variables proporcionadas al script por medio de HTTP cookies. Es equivalente a lo que antes se conocía como `$HTTP_COOKIE_VARS`.

`$_FILES`

Variables proporcionadas al script por medio de la subida de ficheros via HTTP . Es equivalente a lo que antes se conocía como `$HTTP_POST_FILES`.

`$_ENV`

Variables proporcionadas al script por medio del entorno. Es equivalente a lo que antes se conocía como `$HTTP_ENV_VARS`.

`$_REQUEST`

Variables proporcionadas al script por medio de cualquier mecanismo de entrada del usuario. 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.

Nota: Como en `$_REQUEST` se pueden mezclar variables que pueden venir de varios lugares, no suele ser muy recomendable usarla, porque nos pueden inyectar datos de diversas maneras. Si los datos sabemos que nos llegan de un formulario lo suyo es usar `$_POST`, que nos asegura que ninguna entrada por otro lugar nos contaminará ese conjunto de variables. Solo podrías confiar en `$_REQUEST` si la operación que quieres realizar es realmente poco crítica para la seguridad de tu aplicación.

`$_SESSION`

Variables registradas en la sesión del script. Es equivalente a lo que antes se conocía como `$HTTP_SESSION_VARS`. Vea también la sección Funciones para el manejo de sesiones para más información.

Conclusión

De momento dejamos este conocimiento en el aire. Abordaremos cada una de estas variables superglobales en el futuro, en diversas partes del manual. Por ejemplo `$_SESSION` lo veremos cuando hablemos de la sesión en PHP, `$_POST` cuando se explique el envío de formularios, etc.

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en *01/01/2001*

Disponible online en <http://desarrolloweb.com/articulos/12.php>

Operadores en PHP

Lista descriptiva de los operadores más frecuentemente utilizados

Para avanzar nuestro conocimiento en el lenguaje PHP tenemos que detenernos a explicar los operadores, que son los símbolos que nos permiten expresar todo tipo de operaciones entre datos.

Cuando se estudia un lenguaje de programación, aprender los operadores es algo que generalmente realizas sobre la marcha, es decir, a medida que vas haciendo ejemplos vas aplicando operadores y su memorización es algo que surge de manera natural, sin tener que realizar muchos esfuerzos.

Es por tanto, que en el [Manual de PHP](#) nos vamos a detener a comentar cuáles son los operadores en PHP, pero las prácticas vendrán poco a poco a lo largo de sucesivos artículos. No pretendas memorizarlos todos, simplemente tenlos en cuenta y vuelve aquí más adelante para refrescar el conocimiento.

También advertimos que los operadores en PHP son más ricos de lo que se va a ver en este artículo, es decir, existen más operadores que no vamos a incluir en este artículo. Sin embargo, hay que aclarar que en el 99.9% de las ocasiones que usas operadores serán siempre los que vamos a conocer a continuación.

Qué son operadores

Un operador nos permite realizar una operación entre uno o más valores. El operador toma esos valores de entrada y los relaciona entre si, realizando una operación y aplicando otro valor como resultado. Para entendernos, operadores son los símbolos que usamos en las matemáticas para expresar cuentas con números, como los símbolos que nos indican ciertas operaciones: suma, resta, multiplicación, división...

En un lenguaje de programación los valores que vamos a relacionar mediante operadores pueden estar contenidos en variables, o escritos en el propio código. Los operadores toman esos valores y los modifican generando nuevos valores. A veces son conocidas operaciones matemáticas, pero a veces son operaciones lógicas o de asignación, por poner varios ejemplos.

Operadores en PHP por tipos de operaciones

En los capítulos anteriores ya hemos utilizado en nuestros ejemplos algunos operadores. Ahora les vamos a poner nombres, agrupados atendiendo al tipo de operación que realizan.

Operadores aritméticos

Nos permiten realizar operaciones numéricas con nuestras variables. Son los más fáciles de asimilar, porque generalmente todos conocemos esas operaciones.

- + Suma
- - Resta
- * Multiplicación
- / División
- % Módulo (resto de la división)
- ** Exponenciación (2 ** 3, elevar 2 la a tercera potencia)

Nota: 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](#).

Operadores de asignación

Los operadores de asignación son los más habituales y nos permiten traspasar valores en variables. Asignar es el proceso por el cual colocamos un valor en una variable.

- = Asignación

En el pasado ya habíamos creado variables y asignado valores, por lo que lo debes de reconocer. Lo usamos así:

```
$valor = 'Esto es lo que se va a asignar a la variable';
```

Además, en la asignación podría darse el caso que el valor a asignar fuese el resultado de realizar una operación.

```
$a = 23;  
$b = 3;  
$c = $a - $b;
```

Nota: Después de la ejecución de ese código ¿Cuál será el valor de la variable \$c?. Seguro que deduces el valor 20. Si es así estás en lo cierto. El resultado de la operación de \$a (que vale 23) menos \$b (que vale 3) es igual a 20, que se asigna todo seguido a la variable \$c. Lo que has visto como valor a asignar a \$c (\$a - \$b) es lo que se conoce como una expresión. Antes de asignar se realiza el cómputo de la expresión y luego se asigna el valor.

Como has podido ver, asignar es una operación bien simple, lo que está a la derecha del operador se asigna a la variable que está a la izquierda. No tiene más. Sin embargo, PHP incluye una serie de operadores de asignación con una operación asociada, que nos permiten realizar esa operación y una asignación en un único paso.

- += Suma y asignación

- -= Resta y asignación
- *= Multiplicación y asignación
- /= División y asignación
- %= Módulo y asignación
- .= Concatenación y asignación

```
$x = 5;
$x += 5; // $x valdrá 10

$cadena = "Hola ";
$cadena .= "mundo"; // $cadena ahora vale "hola mundo"
```

Nota: Mira más abajo "operaciones sobre cadenas" para entender la concatenación.

Operadores de comparación

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

- == Comprueba si son iguales
- != Comprueba si son distintos
- === Comprueba si son iguales y de exactamente el mismo tipo
- !== Comprueba si son distintos o de distinto tipo
- <> Diferente (igual que !=)
- < Menor que, comprueba si un valor es menor que otro
- > Mayor que
- <= Menor o igual
- >= Mayor o igual
- <=> Comparador de orden. (PHP 7)
- ?? uno o el otro (PHP 7)

En este conjunto de operadores de comparación encontramos varios operadores nuevos, incorporados en PHP 7. Los estudiaremos aparte. Pero los que más se utilizan son los otros, que realmente son bastante sencillos de aprender, aunque para ver ejemplos interesantes tenemos que entender las estructuras de control.

Vamos a adelantarnos un poco, presentando aquí un código en el que usamos la estructura de control condicional de PHP, que nos servirá para comparar valores dados en una expresión y hacer cosas cuando cumpla o no cierta condición.

```
$a = 20;
$b = 30;

if($a < $b) {
```

```
echo '$a es menor que $b';  
}
```

Ejecutando ese código obtendremos la salida "\$a es menor que \$b", porque la operación de comparación se realizó y su resultado fue afirmativo.

Es interesante, aunque incidiremos más adelante, la existencia de los operadores `===` y `!==` que no solo comprueban si algo es igual a otra cosa, sino que además tienen en cuenta los tipos de las variables. Mira primero este código:

```
$a = 20;  
$b = "20";  
  
if($a == $b) {  
    echo '$a es igual que $b'; // (sin tener en cuenta los tipos)  
}
```

Como resultado de ejecución PHP nos dirá "\$a es igual que \$b". Esto es porque la operación de comparación realizada con el operador `==` no tiene en cuenta los tipos de las variables. Para PHP `20` y `"20"` es lo mismo. Sin embargo, ahora mira el siguiente código:

```
$a = 20;  
$b = "20";  
  
if($a === $b) {  
    echo '$a es igual que $b, y del mismo tipo';  
} else {  
    echo '$a es distinto que $b, o de distinto tipo';  
}
```

Al ejecutar ese código obtendremos la salida "\$a es distinto que \$b, o de distinto tipo". Porque al usar el operador `===` sí le estamos pidiendo a PHP que tenga en cuenta los tipos para decirnos si algo es igual a otra cosa.

Nota: Puedes saber más sobre la estructura "if" en el artículo sobre la [estructura condicional de PHP](#).

Operadores lógicos

Los operadores lógicos sirven para realizar operaciones lógicas, valga la redundancia. Son operaciones que al final van a devolver un "sí" o un "no", positivo o negativo. Se usan en combinación con los operadores de comparación cuando la expresión de la condición lo requiere.

- `and` Operación lógica "y", será verdadero si ambos son verdaderos.
- `or` Operación lógica "o", será verdadero si uno de ellos es verdadero
- `xor` Operación lógica "xor", será verdadero si uno de ellos es verdadero, pero no ambos.
- `!` Operación de negación, si era verdadero pasa a falso y viceversa.
- `&&` Operación lógica "y"
- `||` Operación lógica "o"

Los operadores lógicos también se usan generalmente en expresiones que vas a usar para evaluar en estructuras de control. Su operación tiene sentido generalmente cuando trabajamos con valores booleanos (sí o no) o expresiones que una vez evaluadas producen valores booleanos.

```
$a = true;
$b = false;

if($a && $b) {
    echo '$a y $b son los dos verdaderos';
} else {
    echo 'o uno de $a o $b son falsos, o los dos son falsos';
}
```

Ese código nos dirá que "o uno de \$a o \$b son falsos, o los dos son falsos".

Nota: Puede parecer curioso que en PHP haya dos operadores que sirven para la operación lógica "y" y dos para la expresión lógica "o". De hecho es bastante extraño. El motivo es que cada operador tiene una precedencia de operadores distinta.

Operadores de incremento

Sirven para aumentar o disminuir en una unidad el valor de una variable. Son atajos para hacer una operación típica en los recorridos de bucles, de aumentar o disminuir un valor que usamos para llevar la cuenta de las iteraciones realizadas.

- `++$a` Pre-incremento
- `$a++` Post-incremento
- `--$a` Pre-decremento
- `$a--` Post-decremento

Estos operadores son interesantes, porque realizan dos cosas, igual que los operadores de asignación combinada que vimos antes. Por un lado un decremento o incremento y por otro lado se devuelven ellos mismos como resultado del operador. Lo curioso es que existen los "pre" y los "post" y es que funcionan de manera un poco distinta.

Cuando tenemos un pre-incremento o pre-decremento, primero se realiza el incremento/decremento y luego se devuelve ese valor. Con el post-incremento o post-decremento, primero se devuelve el valor original y luego se realiza el incremento/decremento.

Para acabar de entenderlo es mejor examinar este código:

```
$a = 3;
$b = ++$a;
echo "\$a vale $a y \$b vale $b"; // $a vale 4 y $b vale 4

$a = 3;
$b = $a++;
echo "\$a vale $a y \$b vale $b"; // $a vale 4 y $b vale 3
```

En este código como salida obtenemos cosas distintas por el pre-incremento y el post-incremento. La salida aparece al lado, en el comentario. Analízalo y podrás entender cómo es posible.

Operadores de cadenas

Cuando trabajamos con cadenas de caracteres tenemos un operador especial que es el de la concatenación. Sirve para unir una cadena a la otra.

- `.` Concatenación

```
$saludo = "Hola ";
$nombre = "DesarrolloWeb.com";

$saludoCompleto = $saludo . $nombre; // vale "Hola DesarrolloWeb.com"
```

Precedencia de operadores

Cuando se aprende un lenguaje de programación debemos prestar atención especial a la precedencia de operadores, que es básicamente un conjunto de reglas y orden por el que se irán evaluando los operadores cuando trabajamos con expresiones que incluyen varios de ellos.

Igual que en las matemáticas, en ocasiones, si hacemos las operaciones en un orden u otro, los valores finales pueden también tener cambios. En lenguajes de programación, dependiendo del orden en el que se apliquen esos operadores, las expresiones pueden dar pie a resultados distintos.

Por ejemplo, toma la expresión: $2 * 3 + 10$

- Si se resuelve primero la suma sería $2 * 13 = 26$.
- Si se resuelve primero la multiplicación sería $6 + 10 = 16$.

¿cuál de esas dos opciones será la que PHP da por válida? La precedencia de operadores sirve para aclarar esa duda y que los programadores sepan a priori cómo el lenguaje va a resolver las expresiones, siendo capaces de escribirlas de modo que la solución sea la que ellos esperaban.

Sin embargo, no siempre la precedencia del lenguaje es la que nosotros deseamos que se

aplique. Para facilitar las cosas existen los paréntesis, que nos permiten definir nuestras expresiones marcando qué operadores deben resolverse antes. Dicho de otro modo, siempre que usemos paréntesis estaremos obligando al compilador a resolver antes determinadas operaciones, a pesar de la precedencia que él tenga definida de manera predeterminada.

La expresión de antes, escrita de este modo: $(2 * 3) + 10$ no tendría lugar a distintas interpretaciones. Obviamente, debemos conocer la precedencia de operadores para no dar lugar a casos donde el intérprete de PHP pueda obtener resultados no esperados. O si no, estaremos obligados a usar siempre paréntesis, produciendo expresiones de complejidad superior a la necesaria.

Nota: También puede darse el caso que, aunque sepamos bien la precedencia, colocar unos paréntesis puede ayudar a la legibilidad del código, dado que cualquier lector humano, al ver los paréntesis sabrá inmediatamente que ese pedazo de expresión se evaluará antes.

Te recomendamos leer la documentación de PHP para [aprender más sobre la precedencia](#) y consultar la tabla completa, que es un poco larga como para reflejarla aquí.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 12/12/2016
Disponible online en <http://desarrolloweb.com/articulos/operadores-php.html>

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.

Tablas o Arrays en PHP

Creación de tablas por medio de variables tipo array. Utilidad de los arrays en lenguajes de programación en general y en PHP en particular, junto con algunas funciones relacionadas.

Un tipo de variable que ya hemos descrito pero puede ser relativamente complicado a asimilar, con respecto a la mayoría, son los arrays. En éste y otra serie de artículos del Manual de PHP vamos a abordar qué son los Arrays y cómo usarlos en el lenguaje PHP.

Como has podido entender antes, una variable generalmente almacena un dato, ya sea de tipo cadena, numérico, etc. Bueno, pues un array es como una variable capaz de almacenar un conjunto de datos. También los podemos conocer con el nombre de "arreglo", "tabla" o "matriz".

Dado que en array somos capaces de almacenar varios elementos, es necesario el uso de un índice para poder referirnos a cada uno de ellos. Ese índice a veces se conoce como "clave". Existen en PHP arrays con índices numéricos (los arrays más comunes) y con índices alfanuméricos (también llamados arrays asociativos, muy útiles, pero menos comunes), que veremos también en este artículo.

Arrays comunes, índices numéricos

En 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, comúnmente llamados valores, por números. Los números del 1 al 5 son por lo tanto las claves y los sentidos ("tocar", "oir"...) son los valores asociados.

Arrays asociativos

Si lo deseamos, es posible emplear nombres (cadenas) para clasificar los elementos del array. Lo único que deberemos hacer es entrecomillar las llaves alfanuméricas y entonces tendremos un array asociativo:

```
$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");
?>
```

Arrays multidimensionales

Una forma muy practica de almacenar datos es mediante la creación de arrays multidimensionales (tablas o matrices con más de una dimensión). 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"
?>
```

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 del lado izquierdo (indentar) unas líneas más que otras. 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.

La utilidad de los arrays en lenguajes de programación es enorme. Con ellos se resuelven todo tipo de necesidades: contar con estructuras de datos que nos permitan realizar determinados tipos de acciones y realizar [algoritmos](#) capaces de resolver de una manera elegante la más variada gama de procedimientos.

Funciones de Array en PHP

PHP incluye un nutrido conjunto de funciones para trabajar con Arrays. En ellas nos podemos apoyar para realizar toda una serie de operaciones típicas como ordenar elementos por orden alfabético directo o inverso, por claves, contar el numero 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
<code>array_values(mi_array)</code>	Lista los valores contenidos en <code>mi_array</code>
<code>asort(mi_array)</code> y <code>arsort(mi_array)</code>	Ordena por orden alfabético directo o inverso en función de los valores
<code>count(mi_array)</code>	Nos da el numero de elementos de nuestro array
<code>ksort(mi_array)</code> y <code>krsort(mi_array)</code>	Ordena por orden alfabético directo o inverso en función de las claves
<code>list(\$variable1, \$variable2...)=mi_array</code>	Asigna cada una variable a cada uno de los valores del array
<code>next(mi_array)</code> , <code>prev(mi_array)</code> , <code>reset(mi_array)</code> y <code>end(mi_array)</code>	Nos permiten movernos por dentro del array con un puntero hacia delante, atrás y al principio y al final.
<code>each(mi_array)</code>	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](#). Más adelante, cuando leas sobre las estructuras de

control en el [Manual de PHP](#), también encontrarás diversos ejemplos de recorridos a arrays.

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 12/12/2016
Disponible online en <http://desarrolloweb.com/articulos/arrays-en-php.html>

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 código de ejemplo de uso.

En el artículo anterior, sobre [Arrays en PHP](#), explicamos cómo era la sintaxis para la creación de este tipo de estructuras de datos en PHP. Ahora que ya sabes cómo crear arrays, tanto con índices numéricos como con índices asociativos, vamos a ver varios ejemplos de trabajo en PHP que ilustrarán un poco el funcionamiento de algunas de las funciones de arrays (arreglos, vectores, matrices o tablas en castellano).

Este artículo además te servirá para conocer y aprender a usar algunas de las funciones más útiles que trae consigo PHP para el trabajo y manipulación de arrays. Tenemos que advertir que, si estás comenzando con PHP leyendo el [Manual de PHP](#) de DesarrolloWeb.com, algunos de los ejemplos verás que tienen código con elementos del lenguaje que no hemos tocado todavía, como las estructuras de control para hacer bucles con los que recorrer los elementos de un arreglo. Esperamos que no te despiste. Recuerda que más adelante en este manual podrás encontrar explicaciones sobre todo ello. Dicho eso, vamos sin más a introducirnos en materia con varios ejemplos interesantes.

Los ejemplos que podrás encontrar están divididos en dos secciones, aumentar el número de posiciones de un array o reducir el número de casillas disponibles.

Reducir el tamaño de un array

Con las siguientes funciones consigues quitar elementos en un array. Veremos cómo retirar casillas de arrays de tres maneras distintas, pero recuerda que tienes muchas más alternativas si lees la documentación de [funciones para trabajo con Arrays](#).

`array_slice()`

Para disminuir el número de casillas de un arreglo tenemos varias funciones. Entre ellas, `array_slice()` la utilizamos cuando queremos 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).

```
<?php
$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.

```
<?php
$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.

```
<?php
$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

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.

```
<?php
$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()

```
<?php
$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.

Referencia: en el taller de PHP tenemos artículos sobre el trabajo con arrays en PHP. Entre otros encontrarás uno que trata un tema de frecuente consulta: los distintos tipos de ordenación de arrays en PHP: [Ordenar arrays con PHP](#)

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en *09/10/2003*
Disponible online en <http://desarrolloweb.com/articulos/funciones-para-trabajo-arrays-php.html>

Cadenas o strings en PHP

Aspectos relevantes de las cadenas o strings en PHP, un tipo de variables muy utilizado. Lista de caracteres protegidos.

Uno de los tipos de datos más corrientes en la mayoría de los lenguajes son los strings. También podremos conocerlas con el nombre de cadenas o "cadenas de caracteres". No son más que información que contiene texto, con caracteres alfanuméricos, cualquier mezcla de caracteres alfabéticos, símbolos y caracteres numéricos.

Por decirlo con otras palabras, en general, cuando guardamos cualquier texto en una variable, lo que estamos guardando técnicamente son cadenas o strings. Es un tipo de datos muy importante, ya que mucha de la información con la que tenemos que trabajar en las aplicaciones es texto.

Para asignar a una variable un contenido de tipo cadena, lo escribiremos entre comillas, valiendo tanto las comillas dobles como las comillas simples. En código da lugar a sentencias de este tipo:

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

Si queremos mostrar en pantalla el valor de una variable o bien un mensaje cualquiera usaremos la instrucción *echo* como ya lo hemos visto en ejemplos diversos hasta el momento en el [Manual de PHP](#):

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

Nota: En PHP existen diversos mecanismos para producir salida (escribir texto en la página web). La instrucción "echo" es el más sencillo que existe, válido cuando estás dentro de código PHP.

A la sentencia echo le podemos pasar no solo una variable de tipo cadena, pues en realidad saca por pantalla cualquier cosa: Si no es una cadena hará lo que pueda para producir una salida adecuada. Incluso podemos pasarle un literal de cadena:

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

Nota: en lenguajes de programación en general, un "literal" se refiere a un dato escrito tal cual en el código. Un literal de tipo cadena se escribe entre comillas, pero un literal numérico se escribe sin las comillas.

Literales de cadena con comillas dobles o comillas simples

Algo característico de PHP es que permite usar tanto comillas simples como comillas dobles y, dependiendo de cómo lo hayamos hecho PHP interpretará las cadenas de manera distinta. Es algo que debemos de aprender cuanto antes al trabajar en PHP, porque si no, puede que nos de algunos problemas o surjan situaciones en las que el resultado de un programa no sea el que se esperaba.

Cadenas con comillas dobles

Si usamos comillas dobles para delimitar cadenas de PHP haremos que el lenguaje se comporte de una manera más "inteligente". Lo más destacado es que las variables que coloquemos dentro de las cadenas se sustituirán por los valores. Es mejor verlo con un código.

```
$sitioweb = "DesarrolloWeb";  
$cadena = "Bienvenidos a $sitioweb";  
echo $cadena;
```

Ese código producirá como salida "Bienvenidos a DesarrolloWeb". Es decir, PHP interpolará en la variable \$cadena el valor de la variable \$sitioweb, sustituyendo \$sitioweb por su correspondiente valor: "DesarrolloWeb".

Dentro de las cadenas delimitadas por comillas dobles hay una gran cantidad de caracteres de escape, mediante los cuales podemos colocar en cadenas de caracteres cosas como saltos de línea, tabuladores o símbolos "[[--body--]]quot; que no serían considerados como inicio del nombre de una variable. Luego daremos más detalle sobre esto.

Cadenas con comillas simples

Cuando encierras un literal de cadena con comillas simples la cosa cambia bastante. Lo más destacable es que ninguna de tus variables se sustituirá por su valor. Puedes verlo en el siguiente código fuente:

```
$sitioweb = 'DesarrolloWeb';  
$cadena = 'Bienvenidos a $sitioweb';  
echo $cadena;
```

Este código fuente es prácticamente igual que el anterior, con la salvedad que estamos usando cadenas delimitadas por comillas simples. La salida es sensiblemente distinta, en este caso nos mostraría "Bienvenidos a \$sitioweb", dado que no realiza la interpolación de la variable.

Como puedes ver, dentro de una cadena indicada con comillas simples no puedes insertar valores de variables de manera tan sencilla, sino que tendrías que romper la cadena y concatenar con la variable. En seguida hablaremos de cómo concatenar o unir cadenas, pero para que quede ya el ejemplo, para conseguir el resultado del script equivalente pero con comillas dobles, tendrías que escribir algo como esto:

```
$sitioweb = 'DesarrolloWeb';  
$cadena = 'Bienvenidos a ' . $sitioweb;  
echo $cadena;
```

¿Qué usar, comillas simples o dobles?

La respuesta es bien sencilla. Por lo general se recomienda usar comillas simples, puesto que a PHP le costará menos usarlas, dado que no intentará sustituir valores de variables dentro de ellas. Solo si quieres beneficiarte de las ventajas de la interpolación sencilla de variables sería recomendable usar las cadenas con comillas dobles, dado que generarás un código mucho más sencillo de leer por los humanos y de mantener durante la vida de la aplicación.

Concatenación de cadenas

Podemos yuxtaponer o concatenar varias cadenas utilizando el operador de concatenación de strings, que tiene el símbolo punto ".":

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

Aunque ya lo hemos dicho, usando comillas dobles podrías colocar esas variables dentro de nuestra cadena. Dejamos aquí otro ejemplo:

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

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.

Caracteres de escape

Para incluir el símbolo \$, la contrabarra y otros caracteres utilizados por el lenguaje dentro de las cadenas y no confundirlos se usan los caracteres de escape.

Para insertar un caracter de escape tenemos que indicarlo comenzando con el símbolo de la contrabarra (barra invertida) y luego el del caracter de escape que deseemos usar.

Los caracteres de escape disponibles dependen del tipo de literal de cadena que estemos usando. En el **caso de las cadenas con comillas dobles** se permiten muchos más caracteres de escape. Los encuentras en la siguiente tabla:

<code>\n</code>	Salto de línea
<code>\r</code>	Retorno de carro
<code>\t</code>	Tabulador horizontal
<code>\v</code>	Tabulador vertical (incluido en PHP 5.2.5)
<code>\e</code>	Tecla de escape (incluido en PHP 5.4.4)
<code>\f</code>	Tecla de avance de página (incluido en PHP 5.2.5)
<code>\\</code>	Contrabarra o barra invertida
<code>\\$</code>	Símbolo de dólar
<code>\"</code>	Comillas dobles

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 "`
`" y no `\n`.

Nota: El caracter de escape de salto de línea `\n` sólo cambia de línea en el código HTML creado y enviado al navegador cuando la página es ejecutada en el servidor. Ese salto de línea no tiene valor en el HTML, por lo que solamente lo verías al examinar el código fuente producido al ejecutar el script.

En el **caso de las cadenas expresadas con comillas simples** hay muchos menos caracteres de escape. Primero porque no son necesarios (como el símbolo \$, que no puede ser confundido con el inicio de una variable, ya que no las tiene en cuenta) y segundo porque simplemente no se encuentran disponibles.

A continuación puedes ver la tabla de caracteres de escape permitidos en una cadena encerrada mediante comillas simples:

<code>\'</code>	Comillas simples
<code>\\</code>	Contrabarra o barra invertida

Sintaxis compleja de las llaves

Otra cosa útil que nos conviene aprender cuando trabajamos con cadenas es la posibilidad de interpolar valores complejos de variables un tanto especiales. En la documentación de PHP le llaman a esto la "sintaxis compleja", pero no conviene asustarse porque en realidad es bien simple.

Mira el siguiente código:

```
$array = array(1, 2, 40, 55);
$cadena = "La posición tres contiene el dato $array[2]";
echo $cadena; //escribe La posición tres contiene el dato 55
```

Aquí no surge ningún problema al expandir el valor de la posición 3 del array en la cadena, usando (eso sí) comillas dobles. Incluso aunque el array necesite un índice, PHP sabe que lo que tiene que mostrar ahí es una casilla del array. Pero ahora observa el siguiente código:

```
$array = array('uno' => 1, 'dos' => 2, 'tres' => 40, 'cuatro' => 55);
$cadena = "La posición 'tres' contiene el dato $array['tres']"; //esto produce un error!!
```

En este caso nuestro script producirá un error al ser interpretado por PHP, puesto que un array con índice alfanumérico (array asociativo) no es capaz de procesarlo bien cuando lo escribimos dentro de una cadena.

Para salvar esta situación entran en juego la mencionada sintaxis compleja de las llaves. Simplemente vamos a escribir el array asociativo que deseamos que PHP sustituya encerrado entre llaves. Así PHP lo reconocerá perfectamente.

```
$array = array('uno' => 1, 'dos' => 2, 'tres' => 40, 'cuatro' => 55);
$cadena = "La posición 'tres' contiene el dato ${array['tres']}"; //Ahora funciona bien
echo $cadena; //escribe La posición 'tres' contiene el dato 40
```

Quizás en un primer momento esta sintaxis de las llaves no te parezca muy útil, pero te aseguramos que en tu día a día con PHP la vas a usar bastante, porque muchas veces en PHP tienes datos que te vienen de arrays asociativos, o de otros tipos de estructuras que no se interpolan correctamente cuando estás escribiendo valores dentro de cadenas (siempre con comillas dobles).

Aunque no hemos tratado todavía cómo se reciben datos que te llegan de formularios, podemos adelantar aquí un código sencillo. Imagina que estás recibiendo un formulario y tienes un campo llamado "teléfono" en ese formulario. En la página que recibes ese formulario, te llega como `$_POST['telefono']`. Si quieres colocar ese teléfono dentro de una cadena podrías usar un código como este:

```
$telefonoPrefijo = "(+34) ${$_POST['telefono']}";
```

Funciones de cadenas

Las cadenas pueden asimismo ser tratadas por medio de funciones de todo tipo. PHP es un lenguaje muy rico en este sentido, que incluye muchas posibles acciones que podemos realizar sobre ellas con tan solo ejecutar una función: Dividir las en palabras, eliminar espacios sobrantes, localizar secuencias, reemplazar caracteres especiales por su correspondiente en HTML, etc.

Por ejemplo aquí puedes ver el uso de una función bastante útil al programar en PHP y producir salida en HTML, en la que cambiamos todos los caracteres especiales de las entidades HTML (útil para evitar que se inyecte código HTML al documento que no queremos que aparezca formateado, sino escrito en la página con sus etiquetas y todo).

```
$cadenaOriginal = '<b>Me gusta PHP</b>';  
$cadenaRetocada = htmlspecialchars($cadenaOriginal);  
echo $cadenaRetocada; //escribe &lt;b&gt;Me gusta PHP&lt;/b&gt;
```

Más adelante veremos algunos nuevos ejemplos de funciones de cadenas. Pero como siempre recomendamos, recuerda mantenerte informado también con la documentación: [funciones de string](#).

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 14/12/2016
Disponible online en <http://desarrolloweb.com/articulos/cadenas-strings-php.html>

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.

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 podemos invocar las veces que haga falta. Ya sabemos por tanto que las funciones pueden recibir parámetros, que son como variables dentro de la función a las que se le asigna valores en el momento de su invocación. Las funciones pueden servir para realizar tareas sencillas o complejas y como programadores es uno de las primeras herramientas que debemos de conocer para poder estructurar el código de un programa.

PHP basa su eficacia principalmente en su enorme biblioteca de funciones. Una gran librería que crece constantemente, a medida que nuevas versiones van surgiendo y se van incorporando nuevas áreas de trabajo dentro del lenguaje. Las funciones de PHP nos permiten realizar de una manera sencilla tareas habituales y a la hora de desarrollar una aplicación, pero además nosotros podemos hacer nuevas funciones para resolver todo tipo de tareas más específicas de nuestra aplicación.

Las funciones integradas en PHP son muy fáciles de utilizar y a lo largo de todo el [Manual de PHP Básico](#) y otros manuales de Desarrolloweb.com iremos repasando las más usadas. Para acceder a todas las utilidades que hay detrás de una función tan sólo hemos de realizar la llamada (o invocación) de la forma apropiada y especificar los parámetros necesarios para que la función realice su tarea.

Nota: Después de la llegada de PHP 5, en el momento en el que PHP pasó a ser un lenguaje con una orientación a objetos potente, las funciones de la biblioteca del lenguaje tienen en muchos casos alternativas en base a clases y objetos. Nosotros como programadores podemos escoger trabajar con funciones corrientes o con métodos de objetos y clases, pero en general la funcionalidad a la que llegaremos es exactamente la misma.

Crear nuestras propias funciones en PHP

Lo que puede parecer ligeramente más complicado, pero que con un mínimo de experiencia resultará muy sencillo y 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 calculo 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 las 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 interfaces de la web como podrían ser barras de navegación, formularios de login, etc.

Como has podido comprobar, para crear una función debemos declararla. Para ello usamos la palabra `function` seguida del nombre de la función. Luego unos paréntesis donde podemos indicar los parámetros que se espera recibir en su invocación y finalmente el bloque de código de la función propiamente dicha, encerrado entre llaves. En los siguientes artículos seguiremos hablando de los componentes de una función y viendo diversos ejemplos, así que no debes preocuparte si todavía no lo ves demasiado claro.

Estructurar el código de una aplicación con nuestras propias librerías de funciones

Por supuesto, la función ha de ser definida para poder ser utilizada, ya que no se encuentra integrada en PHP sino que la hemos creado nosotros. Si pensamos que en una aplicación web completa podemos tener cientos de funciones definidas por nosotros mismos quizás te asuste que tengas demasiado código de funciones que deben ser definidas antes de ser usadas. Pero esto en realidad no pone ninguna pega, ya que pueden ser incluidas desde un archivo externo. De hecho es muy común que tengamos archivos donde solo colocamos el código de las funciones, almacenando definiciones de las funciones que vayamos creando para realizar un

sitio web.

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

```
require("ruta/a/libreria.php");
```

O si prefieres la alternativa del `include`:

```
include("ruta/a/libreria.php");
```

Nota: Tanto `require()` como `include()` hacen el mismo trabajo, de traerse código que hay en archivos diferentes dentro del servidor, para que podamos utilizarlo al crear una página. La diferencia fundamental entre `require` e `include` es que la primera requiere forzosamente algo y la otra no. Es decir, si hacemos un `require()` de un archivo y éste no se encuentra disponible por cualquier motivo, PHP parará la ejecución del código y devolverá un "Error fatal". Si por el contrario hacemos un `include()` y el archivo que tratamos de traer no se encuentra disponible, entonces lo que PHP nos mostrará es una señal de advertencia, un "warning", pero tratará de seguir ejecutando el programa.

En resumen, cuando usas archivos con código de funciones (librerías) y los incluyes para usarlos desde otras páginas de la aplicación, la cosa quedaría así:

Tendríamos un archivo `libreria.php` como sigue

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

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 reutilizar algunas de nuestras librerías 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.

Nota: 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 de código y dedicación podemos llegar a poseer nuestro pequeño tesoro de funciones, capaces de aligerar nuestro trabajo diario.

De hecho, más adelante si sigues aprendiendo PHP profesionalmente encontrarás que existen los frameworks, que son en cierto modo como bibliotecas adicionales de funciones que puedes usar para resolver muchas más cosas, adicionales a las que el propio lenguaje ya te ofrece. Los frameworks además ayudan a los programadores a estructurar su código y a usar diversos patrones de diseño de software que facilitan la creación de proyectos de fácil mantenimiento y capaces de crecer sin volverse inmanejables. En DesarrolloWeb.com tenemos varios manuales de frameworks PHP.

Como referencia, si deseas puedes repasar todos los conceptos anteriores sobre las funciones, así como diversas otras cosas interesantes, te dejamos enlace hacia el [Videotutorial sobre las funciones en PHP](#).

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-").

Conclusión

Esperamos que esta primera introducción a las funciones en PHP te haya sido aclaradora. En los próximos artículos veremos nuevas cosas importantes para dominarlas, como el paso de parámetros en las funciones y los valores de retorno.

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 04/01/2016
Disponible online en <http://desarrolloweb.com/articulos/12.php>

Paso de parámetros en funciones PHP

Este capítulo explica todos los detalles relativos al paso de parámetros en las funciones PHP.

En el artículo anterior del [Manual de PHP](#) ya comenzamos a explicar las [funciones en PHP](#). Como hemos visto, crear nuestras propias funciones no es complicado, pero tenemos que aprender diversas cosas nuevas para extraer toda su potencia.

Ahora vamos a explicar algunos detalles adicionales sobre la definición y uso de funciones en PHP, para ampliar la información anterior. En concreto, hablaremos sobre los parámetros en las funciones, ya que hay mucho más que decir para abarcar todas las posibilidades de PHP: el paso de parámetros por valor, paso por referencia, los valores predeterminados, etc. Además en este artículo veremos nuevos ejemplos de funciones que nos sirvan para ir practicando con nuevos ejemplos en PHP.

Paso de parámetros

Los parámetros son los datos que reciben las funciones y que utilizan para realizar las operaciones de esa función. Una función puede recibir cualquier número de parámetros, incluso ninguno.

Si la función que estamos construyendo no necesita recibir ningún parámetro, al declararla, simplemente indicamos los paréntesis vacíos en la cabecera. Por ejemplo en la siguiente función mostramos la fecha del día de hoy. Para ello nos apoyamos en otra función incluida en PHP: `date()`.

```
function fecha_hoy() {  
    echo date('d/m/Y');  
}
```

La intención de la anterior función es mostrar la fecha del día actual. Como siempre mostrará el día de hoy, no necesito pasarle ningún parámetro, siempre hará lo mismo. Las funciones que no requieren parámetros se las invoca indicando los paréntesis vacíos.

```
fecha_hoy();
```

Nota: Si quieres saber más sobre la función `date()` y otras maneras de crear fechas en PHP, te recomendamos el artículo [Crear y convertir fechas en PHP](#)

En el caso que queramos, o necesitemos, recibir parámetros para implementar una función, a la hora de definirla, en la cabecera, se definen los parámetros que va a recibir.

```
function f1 ($parametro1, $parametro2)
```

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. Es decir, la función necesitará que le enviemos dos datos, pero no le importará que sean de un tipo u otro.

Los parámetros tienen validez durante la ejecución de la función. Se dice que 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
```

Ten en cuenta que, en el caso que quieras usar parámetros con valores por defecto, estás obligado a que éstos se declaren al final en la lista de parámetros de la cabecera de la función.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 04/01/2017
Disponible online en <http://desarrolloweb.com/articulos/parametros-funciones-php.html>

Retorno de valores en funciones PHP

Cómo devolver valores en funciones PHP, con la palabra return. Explicamos varias formas de realizarlo con nuevos ejemplos para aprender a trabajar con funciones en PHP.

Para seguir aprendiendo sobre funciones en el [Manual de PHP](#) necesitamos abordar con detalle el asunto de la devolución de valores en funciones.

Algo que querrás hacer en PHP, y en la programación en general, es crear funciones que, una vez ejecutadas, nos entreguen un valor como resultado. Es algo muy habitual y que implica varias cuestiones que vamos a abordar en el presente artículo.

Las funciones pueden, o no, retornar valores. Es decir, no es obligado que las funciones retornen valor alguno, solo se trata de una posibilidad, que encontrarás de mucha utilidad en el desarrollo en general. De hecho, nuestros anteriores ejemplos de funciones no habían retornado ningún valor y ya habíamos visto que realizaban tareas bastante útiles.

Palabra "return"

Para retornar valores en funciones se utiliza la palabra "return", indicando a continuación el dato o variable que tienen que retornar.

```
function suma($valor1, $valor2) {  
    return $valor1 + $valor2;  
}
```

La anterior función realiza una operación de suma entre dos valores enviados por parámetro. Para invocarla debemos enviarle los dos valores que debe sumar. Cuando se ejecute la función recibiremos un valor como devolución y podremos hacer cualquier cosa con él. Por ejemplo, en el siguiente código estamos invocando a la función suma, enviando dos valores numéricos y almacenando el valor de devolución en una variable llamada "\$resultado".

```
$resultado = suma(3, 6);
```

Una función puede perfectamente tener múltiples palabras return en su código. Sin embargo, aunque esto ocurra, debemos tener en cuenta que una función sólo podrá devolver un único valor. Entre otras cosas esto ocurrirá porque, cuando se usa el return, se termina la ejecución de la función devolviendo el dato indicado.

Observa el siguiente código de función. Realiza una operación de división. Recuerda que la operación matemática de dividir algo entre cero no está permitida, ya que el resultado sería "infinito" y ese valor desbordaría a la máquina. Entonces, antes de realizar la operación de división vamos a hacer una comprobación que no se intente dividir entre cero.

```
function division($valor1, $valor2) {  
    if($valor2 == 0) {  
        return 'No puedo dividir por cero!!';  
    } else {  
        return $valor1 / $valor2;  
    }  
}
```

Nota: En el código anterior hemos usado la estructura de control "if" que veremos más

adelante, en el artículo [Control del flujo en PHP: Condiciones IF](#)

Aunque todavía no hemos explicado qué hace un "if", simplemente es una operación condicional. Comprueba una expresión y hace una cosa u otra dependiendo de lo que valga. En resumen, se evalúa si \$valor2 tiene el dato 0 (cero) y en ese caso se devuelve un mensaje "No puedo dividir por cero!!". En caso que \$valor2 no fuera un valor de cero, entonces realiza la operación de división y se devuelve el resultado.

Apreciarás que en el código de la función anterior aparecen dos `return`, quizás no te resulte extraño, porque solamente uno de ellos se ejecutará, dada la construcción del IF. No obstante, debes saber que un `return` siempre detiene en ese punto la ejecución de la función. Es decir, después de ejecutar un `return` no se ejecutará ninguna otra línea de código siguiente.

Para que quede claro, ahora mira esta otra función.

```
function cuadrado($valor) {  
    return $valor * $valor;  
    echo 'Esto nunca se ejecutará!!';  
}
```

Debido al `return`, el código con la sentencia "echo" nunca se llegará a ejecutar. Así que no aparecerá nunca el mensaje por pantalla, porque el `return` de la línea anterior parará siempre la ejecución de la función.

Para acabar este punto queremos volver sobre la función anterior llamada "division". Dado lo aprendido de las características del `return`, aunque nuestro código fuera ligeramente distinto, también tendríamos el mismo resultado. Observa el siguiente código:

```
function division($valor1, $valor2) {  
    if($valor2 == 0) {  
        return 'No puedo dividir por cero!!';  
    }  
    return $valor1 / $valor2;  
}
```

Aunque no hemos usado la construcción "else" (mira más adelante el capítulo donde hablamos de los if) el segundo `return` solo se ejecutará si \$valor2 es distinto de cero, porque si fuera igual a cero se habría salido de la función debido al primer `return`. Este detalle quizás cuesta un poco de ver al principio de la experiencia como programador, pero es bastante sencillo en realidad. En resumen, puedes confiar que la ejecución de una función siempre se detendrá después de producirse un `return`.

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 y el anterior

sobre [paso de parámetros](#).

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 21%.

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

<body>
<?
function iva($base,$porcentaje=21){
  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 21 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 21% de 1000. En la segunda, se muestra el 7% de mil y en la tercera, el 0% de 10.

Retornar múltiples valores en una función

Lo dicho anteriormente sobre que "una función devuelve un único valor" puede resultar cortante para las personas que están comenzando en la programación, al ver que ello puede significar una gran limitación a la hora de escribir funciones. No obstante, con las herramientas con las que se cuenta en la programación ésto no es así.

Si queremos hacer que se puedan devolver varios valores distintos podríamos que recurrir a un truco que consiste en devolver un array.

```
function numeros_pequenos()
{
  return array (0, 1, 2);
}

list ($zero, $one, $two) = small_numbers();
```

Con el array devuelto podremos hacer cualquier cosa. Acceder a sus casillas por separado,

recorrerlo, etc. Pero en el código anterior hemos hecho mano de una función incorporada en PHP, nueva para ti, llamada `list()`. Ésta 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.

Hay que decir que además de arrays, cuando las cosas se complican también podemos devolver objetos y eso nos ayudará a retornar en las funciones todo tipo de estructuras complejas, con varios datos, solo devolviendo un objeto. Más adelante también hablaremos sobre objetos, así que habrá tiempo de ver ejemplos.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 04/01/2016
Disponible online en <http://desarrolloweb.com/articulos/retorno-valores-return-php.html>

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.

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.

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  
$idioma=substr($_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 (*\$_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](#)

Este artículo es obra de *Rubén Alvarez*

Fue publicado por primera vez en 01/01/2001

Disponible online en <http://desarrolloweb.com/articulos/12.php>

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:

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.

Otro ejemplo del bucle While

El bucle `while` se suele utilizar cuando no se sabe exactamente cuántas 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 caracter "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 caracter "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](#).

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](#).

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 01/01/2001
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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.

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.

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>";}
}
```

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](#).

Este artículo es obra de *Rubén Alvarez*

Fue publicado por primera vez en *01/01/2001*

Disponible online en <http://desarrolloweb.com/articulos/12.php>

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.

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:

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](#)

\$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.

\$_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.

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.](#)

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en *01/01/2001*
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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 interactuar 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](#)

\$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.

\$_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.

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](#).

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en *01/01/2001*
Disponible online en <http://desarrolloweb.com/articulos/12.php>

Conceptos de programación en PHP por la práctica

Vídeo en el que aclaramos y practicamos con conceptos básicos de programación como variables, estructuras de control, bucles, etc.

En este vídeo explicamos algunos conceptos de programación básicos, orientados a personas que no saben programación o que tienen poca experiencia programando. Es la tercera de una serie de tres clases gratuitas impartidas en DesarrolloWeb.com para acercarse al mundo de la programación con PHP.

El objetivo es concretar en líneas de código, y desde un punto de vista práctico, algunos de los conceptos vistos durante las dos primeras clases de PHP para no-programadores, como son las variables, estructuras de control para la toma de decisión y para hacer bucles o repeticiones. Además, queremos realizar algún ejemplo práctico que nos ayude a demostrar lo útil que es PHP para tareas del día a día de los desarrolladores, como es el envío y recepción de un formulario.

Pretendemos acercar el mundo de PHP para personas que no tienen conocimientos de programación de una manera rápida y dar referencias para seguir aprendiendo, así como presentar también la [segunda convocatoria del Curso de PHP](#) que estamos organizando en **EscuelaIT**.

En este artículo encontrarás los códigos de los ejercicios realizados en clase y al final del todo, el vídeo de la propia clase, junto con el turno de preguntas que nos realizaron los estudiantes.

Comenzamos la clase explicando cómo arrancar un servidor web para poder programar con PHP en local, por si alguien todavía anda perdido con ese asunto, y cómo colocar archivos dentro del servidor para poder acceder a ellos desde "localhost". A partir de ello realizamos dos ejemplos:

Conceptos básicos para la programación con PHP

A modo de primeros pasos, y de introducción al mundo de la programación y del lenguaje PHP, en este ejemplo se mostraron en código conceptos básicos de la programación.

1. Comentarios
2. Variables y tipos
3. Operadores, aritméticos, de cadena, comparación, asignación
4. Estructuras de control para toma de decisiones
5. Bucles

El listado de código que se realizó en directo es el siguiente:

```
<!DOCTYPE html>

<html>
<head>
  <title>Ej 1 de PHP</title>
</head>

<body>

<h1>Estoy en PHP!!</h1>
<?php

//codigo PHP
//echo "Hola mundo!!";

// variables
$mivariable = 2323;
$v2 = "texto";

//hacer algo con ellas
echo $v2;
echo "<br>";

//operaciones
// 1+4
// "hola" . "mundo"
// $v3 = "pepe";

/*
  Esto es un comentario
  de muchas
  líneas
*/

$contador = 5 + 5;
$contador = $contador + 10;

echo $contador;

// estructuras de control
```

```
$edad = 34;
if($edad < 18){
echo "Si es menor de 18";
}

// repeticiones o bucles
while($contador < 100){
    // todo esto se repetirá
    echo "contador vale " . $contador;
    echo "<br>";
    $contador = $contador + 7;
}
?>

</body>
</html>
```

Ejemplo práctico de envío y recepción de un formulario

En un segundo ejercicio práctico de esta clase de PHP se mostró lo fácil que es enviar datos desde un formulario a una página web y recibirlos para procesar y realizar acciones diferentes según sus valores.

Se explicó que un formulario se hace en HTML, tal como estamos acostumbrados a hacer si ya hemos realizado páginas web en alguna ocasión. El código de la página con el formulario HTML que realizamos es el siguiente:

```
<!DOCTYPE html>

<html>
<head>
    <title>Formulario</title>
</head>

<body>
    <h1>Ejemplo con formularios</h1>
    <form method="post" action="recibe-formulario.php">
        Nombre: <input type="text" name="nombre" value="Escribe tu nombre">
        <br>
        Edad: <input type="text" name="edad" value="">
        <br>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

Toda la parte de PHP se queda simplemente en la recepción de los datos de ese formulario y su procesamiento, que en este ejemplo se realizó en una página y archivo aparte. El procesamiento que se implementó es muy sencillo, simplemente para darnos cuenta de que básicamente con PHP podemos acceder a los datos del formulario de una manera muy sencilla,

a través de los arrays `$_POST` y mencionando sus "hermanos" `$_GET` o `$_REQUEST`.

El ejemplo es muy simple, aunque poco a poco fuimos complicando un poco para introducir diversos métodos de integrar código PHP en una página HTML, por lo que el listado a continuación puede parecer un poco complejo en un primer vistazo, aunque en la clase se explica todo perfectamente y paso a paso.

```
<!DOCTYPE html>

<html>
<head>
  <meta charset="utf-8" />
  <title>recibo datos del form</title>
</head>

<body>

<?php
//recibo
$nombre_usuario = $_POST["nombre"];
$edad_usuario = $_POST["edad"];

//valido los datos

//echo "Hola " . $nombre_usuario . " tu edad es " . $edad_usuario;

// toma de decisión en función de la edad
if($edad_usuario >= 18){
  ?>
  <div style="color: blue;">
    <b>
<?php
echo "Hola $nombre_usuario, como tu edad es $edad_usuario, puedes ver este contenido!!";
  ?>
  </b></div>
<?php
}else{
  //parte negativa
  echo '<span style="color: red;">';
  echo "Lo siento $nombre_usuario, no puedes ver esta página!!! Adiós!";
  echo "</span>";
}

?>

</body>
</html>
```

Vídeo completo de la tercera clase de PHP para no-programadores

A continuación puedes ver el vídeo completo de la clase en el que se realizan los ejemplos de código anteriores, paso por paso, a lo largo de 90 minutos. Esperamos que os sirva para aclarar

muchos conceptos y también que os resulte entretenido.

Recuerda que puedes seguir aprendiendo PHP con nosotros de manera autodidacta con los manuales gratuitos de DesarrolloWeb.com en la sección [PHP a fondo](#) y que si te interesa aprender de manera intensiva, guiada y personalizada, con clases en vivo durante las próximas semanas, puedes apuntarte con nosotros al [Curso completo de PHP](#).

Para ver este vídeo es necesario visitar el artículo original en:

<http://desarrolloweb.com/articulos/conceptos-programacion-php-practica.html>

Este artículo es obra de *Miguel Angel Alvarez*

Fue publicado por primera vez en 10/09/2013

Disponible online en <http://desarrolloweb.com/articulos/conceptos-programacion-php-practica.html>

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](#).

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 24/04/2004
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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 rellenos 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](#). Además, también te recomendamos ver el [Videotutorial sobre las Cookies en PHP](#).

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en *01/01/2001*
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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.

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 shttp (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](#) o al [Videotutorial sobre las Cookies en PHP](#).

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 19/12/2007
Disponible online en <http://desarrolloweb.com/articulos/cookies-en-php.html>

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.

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 01/01/2001
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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`.

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 se 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 la 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 `_sessionstart()`. 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 `_sessionregister('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](#).

Este artículo es obra de *Rubén Alvarez*

Fue publicado por primera vez en *01/01/2001*
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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.

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 del lado del servidor es el poder trabajar con contenidos que están alojados 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 existentes. Podrás encontrar bases de datos de código abierto, como MySQL, comerciales propietarias como Oracle y además tiene librerías para acceso a datos por ODBC, lo que nos permite comunicar con todas las bases de datos posibles en sistemas Microsoft, como Access o SQL Server. Gracias a los juegos de funciones existentes para cada sistema gestor de base de datos, podremos realizar cualquier acción con los datos que necesitemos para el desarrollo de la más variada gama de aplicación web.

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. A lo largo del manual de PHP nos limitaremos 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 de ese lenguaje de acceso a datos.

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

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

Configurar la base de datos que vamos a utilizar para los ejemplos de este manual

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](#).

Guión de artículos relativos a las bases de datos en PHP

A lo largo del manual ofreceremos varios artículos, en los que trataremos cada uno de los accesos a la base de datos más típicos, como selecciones, actualizaciones o inserciones, siguiendo el siguiente guión:

- [Inserción de registros](#)
- [Selección de registros](#)
- [Actualización de registros](#)
- [Borrado de registros](#)

Para completar todo aquello que debes saber sobre el acceso a bases de datos en PHP encontrarás multitud de artículos prácticos en el [Taller de PHP](#), como son la [Recepción el último ID de una inserción con PHP y MySQL](#), la [Paginación de resultados con PHP y MySQL](#), [Programar un buscador con PHP y MySQL](#), etc.

Además, para los que prefieran verlo todo paso a paso en vivo, contamos con un [videotutorial sobre las bases de datos en PHP](#) que te ayudará a entender mejor cómo trabajar con las bases

de datos en PHP.

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en *01/01/2001*
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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 `_mysqlconnect`. A continuación, deberemos generar una orden de inserción del registro en lenguaje SQL. Esta orden será ejecutada por medio de la función `_mysql_dbquery`. 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>
</HTML>
```

Ejecutar ejemplo

Los parámetros 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 vacías: "".

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, referirse al [tutorial de SQL](#).

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en *01/01/2001*
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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 número 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_fetcharray* 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_freeresult` se encarga de liberar la memoria utilizada para llevar a cabo la consulta. Aunque no es necesaria su utilización, resulta altamente aconsejable.

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 01/01/2001
Disponible online en <http://desarrolloweb.com/articulos/12.php>

Actualización de un registro de la 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_fetcharray` lo que nos permite mostrar cada una de las opciones.

El script de actualización será muy parecido al de inserción:

Nota: Para quien desee una referencia completa sobre cómo se hacen las sentencias UPDATE en el lenguaje SQL, recomendamos leer el artículo [Actualización de registros con UPDATE en SQL](#).

```
<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>
```

Como habéis podido comprobar, la operación UPDATE no tiene demasiada complejidad. Sin embargo, para afianzar estos conocimientos recomendamos ver el [vídeo sobre la actualización de registros en MySQL con PHP](#).

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en 01/01/2001
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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>
```

Nota: Varios comentarios preguntan por el archivo lectura.php que está enlazado en el código del ejemplo. Tenemos que decir que ese archivo se comentó en el artículo de la [Selección y lectura de registros de la base de datos por medio de PHP](#).

Con este capítulo cerramos el bloque de accesos a bases de datos con PHP. Para más información relacionada podéis referiros al [taller de PHP](#) donde podréis encontrar algún que otro artículo interesante al respecto. Asimismo, para los que prefieran el vídeo como material didáctico, recomendamos ver el [Videotutorial sobre las bases de datos con PHP](#).

Este artículo es obra de *Rubén Alvarez*
Fue publicado por primera vez en *01/01/2001*
Disponible online en <http://desarrolloweb.com/articulos/12.php>

Actualización de funciones PHP para acceso a MySQL

Las nuevas versiones de PHP ofrecen juegos de funciones MySQL diferentes de los tradicionales, que implementan diversas mejoras en la gestión de bases de datos.

PHP está en constante evolución y prueba de ello son las novedades que venimos a comentar en este artículo. Se trata de las nuevas funciones de acceso a bases de datos MySQL que están disponibles desde PHP 5, con nombres diferentes a las funciones disponibles en el pasado y que también tienen algunos cambios en el modo de trabajar. Las nuevas funciones están creadas para dar respuesta a diversas mejoras que el sistema gestor de bases de datos MySQL ha incorporado recientemente.



Este es un tema fundamental, sobre todo para las personas que conocen PHP desde hace tiempo, que como yo, estábamos acostumbrados a usar un juego de funciones que a día de hoy se encuentra obsoleto y que no se recomienda utilizar. Realmente, en el momento de escribir este artículo las funciones de MySQL antiguas todavía, se pueden usar, pero la previsión es que se eliminen en PHP 5.5, por lo que resulta imprescindible actualizar nuestros conocimientos.

Afortunadamente, es bastante sencillo el paso a las nuevas funciones de MySQL, puesto que el esquema de funcionamiento es prácticamente el mismo, así como los nuevos nombres de funciones, que tienen muy pocas variaciones. Por lo tanto, si te has leído el [Manual de PHP](#) de DesarrolloWeb.com y has aprendido con nosotros a trabajar con las bases de datos, no tendrás realmente muchos problemas en aprender las novedades que paso a relatar a continuación.

Dos tipos de funciones para acceso a las bases de datos MySQL

Una de las novedades que PHP nos ofrece en las recientes funciones de PHP es la incorporación de dos tipos aproximaciones al trabajo con MySQL, que ahora se puede hacer por procedimientos (funciones normales de toda la vida) y por objetos.

Nota: Antes sólo podíamos acceder a MySQL por funciones comunes, pero ahora también podemos acceder por medio de clases y objetos de programación orientada a objetos (POO). La incorporación en el API de métodos basados en POO hace patente que PHP está teniendo muy presente este paradigma de desarrollo dentro de su núcleo y que la tendencia en las nuevas versiones de PHP será la de potenciarlo aun más.

Funciones basadas en procedimientos

Como venimos haciendo toda la vida con PHP. Funciones normales y corrientes, que reciben parámetros y que devuelven valores.

Funciones basadas en POO

Realmente son constructores para obtener objetos y métodos a los que podemos acceder dentro de esos objetos para realizar las operaciones con la base de datos.

Hay que aclarar que ambas aproximaciones son perfectamente válidas y que no cambian las cosas que se pueden hacer desde una u otra. Por tanto, decantarnos por uno u otro modo no implicará ninguna ventaja en funcionalidad, ni tan siquiera de rendimiento. De hecho, ambos modos de trabajo son el "mismo perro con distinto collar". Por tanto, es tan válida una elección como la otra y nuestra preferencia será más bien por gustos o costumbres a la hora de desarrollar.

Nota: Lo que no se recomienda es en un mismo proyecto cambiar de uno a otro estilo de acceso a base de datos MySQL de manera arbitraria. Debemos tener en cuenta que realmente esa recomendación no es debido a que no te vaya a funcionar tu programa, o puedas tener algún problema determinado, sino más bien es una cuestión de estilo y claridad a la hora de mantener el código de tu proyecto.

Acceso a MySQL con funciones por procedimientos

En el mencionado [Manual de PHP](#) venimos ofreciendo ejemplos para acceso a base de datos con funciones de toda la vida. Por eso, en este artículo vamos a mostrar las diferencias que existen con respecto a las funciones antiguas, usando también código por procedimientos. Sin embargo, como verás a continuación, las funciones son prácticamente las mismas a las que se les ha agregado una "i". Por ejemplo, `mysql_connect()` pasa a ser `mysqli_connect()`. Los parámetros son los mismos en muchos casos, o muy parecidos.

Nota: este artículo no es un tutorial de acceso a bases de datos con PHP, para eso ya disponemos del manual de PHP en DesarrolloWeb.com. Realmente se trata de una serie de guías para poder reciclar nuestros conocimientos con el PHP de versiones anteriores.

Conexión con la base de datos: `mysqli_connect()`

Ahora indicamos también la base de datos a la que queremos conectarnos. Y atención, porque lo que nos devuelve la función no es un *link* o identificador de la conexión con la base de datos, sino un objeto. Pero aunque no sepas POO, no te preocupes por este detalle, pues poco nos

importa a los desarrolladores, dado que vamos a usar ese objeto como si fuera simplemente el identificador de la conexión, o sea, vamos a usarlo como hacíamos anteriormente en PHP.

```
$conexion = mysqli_connect("localhost", "usuario", "clave", "mi_bbdd");
```

Cerrar la conexión: `mysqli_close()`

Tenemos que pasarle como parámetro siempre el objeto que obtuvimos al iniciar la conexión.

```
mysqli_close($conexion);
```

Ejecutar una consulta: `mysqli_query()`

Esta función es exactamente igual, aunque en este caso recibe obligatoriamente como primer parámetro el objeto de conexión a la base de datos que recibimos al conectarnos. El segundo parámetro es la sentencia SQL que se desea ejecutar en MySQL.

```
$resultado = mysqli_query($conexion, "select * from cliente");
```

Otro aspecto interesante es que nos devuelve un valor que puede ser, o bien un *booleano* para saber si la consulta se ejecutó con éxito, como en el caso de un "insert" o "delete", o bien un objeto resultado con un conjunto de registros en el caso de un "select".

Extraer un registro del conjunto de resultados: `mysqli_fetch_array()`

Esta función es exactamente igual, recibe el conjunto de resultados del que queremos sacar un registro. Devuelve en este caso un "array", pero está igualmente `mysqli_fetch_object()` que devolvería un objeto.

```
$fila = mysqli_fetch_array($resultado);
```

Contar el número de filas de un resultado: `mysqli_num_rows()`

También es exactamente igual, simplemente le pasamos el objeto conjunto de registros del que queremos saber el número de filas o registros que contiene.

```
$fila = mysqli_num_rows($resultado)
```

Conclusión

En general, como estás pudiendo comprobar, el sistema no varía mucho de lo que ya conoces en el pasado. Cambian nombres de funciones mínimamente y algún parámetro suelto aquí y allí.

Creo que el paso para usar las nuevas funciones MySQL es tan rápido que no tiene sentido seguir usando las funciones antiguas y además en breve el cambio será obligado porque los accesos antiguos que hacíamos a MySQL dejarán de funcionar.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 11/03/2013
Disponible online en <http://desarrolloweb.com/articulos/actualizacion-funciones-php-mysql.html>

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.

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.

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, 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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 10/06/2003
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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

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,"../");
```

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 10/06/2003
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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, elijeremos 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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 26/06/2003
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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 o, 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:

Si tenemos pocas tablas y bastante sencillas: Las podemos crear en remoto con alguna herramienta como [PhpMyAdmin](#) o [MyCC](#).

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.

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

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 wwwdesarrolloweb.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 inserts

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.

Este artículo es obra de *Miguel Angel Alvarez*
 Fue publicado por primera vez en 26/06/2003
 Disponible online en <http://desarrolloweb.com/articulos/12.php>

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 23/07/2003
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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.

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.

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.

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 terminated by
 - Complete inserts

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.
[Documentation]		
<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.

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 23/07/2003
Disponible online en <http://desarrolloweb.com/articulos/12.php>

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.

Seguridad de proyectos PHP

Cómo evitar que nuestros proyectos web sean atacados fácilmente.

Debido a los avances en nuestro medio tecnológico, nos encontramos con situaciones que nos amargan la existencia, como por ejemplo: cuando nuestras webs se caen, ingreso de virus a nuestro PC, *spam* en nuestros correos, etc.

El término se empezó a utilizar en el año 2000 por Luis von Ahn, Manuel Blum y Nicholas J. Hopper de la Carnegie Mellon University, y John Langford de IBM.

El sistema Captcha tiene las siguientes características por definición:

Son completamente automatizados, es decir, no es necesario ningún tipo de mantenimiento ni de intervención humana para su realización. Esto supone grandes beneficios en cuanto a fiabilidad y coste.

El **algoritmo** utilizado es público. De esta forma, la ruptura de un captcha pasa a ser un problema de **inteligencia artificial** y no la ruptura de un algoritmo secreto.

Pues habiendo visto un poco lo que es **captcha** “**Completely Automated Public Turing test to tell Computers and Humans Apart (Prueba de Turing pública y automática para diferenciar a máquinas y humanos)**).



En este escenario, podríamos encontrarnos con dos situaciones bien diferentes:

- Tener un sitio montado y tener que protegerlo.
- Programar un sitio desde cero.

Escape de las entradas. Para muchos la manera ideal de proteger un *site*. Como ya hemos visto en alguno de los casos, no nos es útil. Los más habituales son el uso de:

- addslashes() / stripslashes()
- htmlentities(\$string, ENT_QUOTES)
- htmlspecialchars()
- mysql_real_string()

Teniendo activadas las **magic_quotes_gpc** en nuestro **php.ini**, que nos pondrá por defecto un **slash** en todos los **strings** (evitando los tediosos "addslashes()"). En todo caso, el uso de dichos elementos nos podrá salvar de muchos de los ataques.

Evitar, salvo en casos necesarios, que los formularios POST se llamen desde otro dominio que no sea el del propio servidor. En este caso, nos evitaremos que un atacante avezado utilice un *script* a tal efecto para ir bloqueando nuestro servidor y llenándolo de datos inútiles.

Vamos a ver, **¿qué clase de configuración sería la óptima para que un sistema PHP fuera más seguro contra todo tipo de ataques?**

Estas directivas serían:

Openbase_dir Esta directiva bien configurada evitará los ataques "trasversal directories", debido a que limita ejecución de ficheros al entorno que escojamos.

Allow_furl_open off Es importante que esta directiva esté en OFF para evitar "Remote File Inclusion", ya que la inhabilitación de esta directiva no permitirá a la aplicación hacer *include* remotos.

Register_globals off Como ya hemos explicado, quizá la más maléfica (y obsoleta) forma de que nuestros atacantes desplieguen todo su potencial es mediante esta directiva activada. Es decir, cualquier parámetro que nos venga por **POST** o **GET** puede ser una variable potencialmente peligrosa en nuestro aplicativo. Así, cualquier parámetro externo se tratará de forma cuidada con **\$_GET['param']**, **\$_POST['param']**, **\$_FILES['param']** para establecer qué tipo de variables son externas y cuáles no.

No se recomienda, a no ser que se tenga muy claro qué se está haciendo, el uso de **\$_REQUEST**, pues ahí puede entrar 'cualquier cosa' que nos venga externamente, y fácilmente podrían introducirnos valores no esperados.

Safe_mode on Esta directiva activada evitará la ejecución de algunos comandos potencialmente dañinos en nuestro sistema, además del chequeo de ciertas funciones que puedan considerarse delicadas. Una lista de dichas funciones puede encontrarse aquí:

php.net/manual/en/features.safe-mode.functions.php

Especial atención merecen también las directivas "safe_mode*" que componen la familia.

- safe_mode:
- safe_mode_gid
- safe_mode_include_dir
- safe_mode_exec_dir
- safe_mode_allowed_env_vars
- safe_mode_protected_env_vars

Por último, unas funciones que, según la casuística de nuestro aplicativo pudiera evitarnos algún susto por la ejecución de comandos sensibles que no queremos (y no debemos) utilizar:

- disable_functions <lista de funciones>
- disable_classes <lista de clases>

Escaneo de puertos Una manera de evitar ataques a todo sistema operativo, ya sea mediante web o mediante cualquier otro tipo de vulnerabilidad, sería mediante la ejecución de código remoto o inyección de código no deseado en servicios que puedan tener relación con nuestro sistema.

Para ello se recomienda ejecutar un escaneo de puertos de nuestra máquina (no únicamente puerto 80-http o 443-SSL) para averiguar las posibles vulnerabilidades o *exploits* que puedan afectar a nuestro sistema y servidor web:

Los más conocidos son **nmap** y **nessus**. El funcionamiento de nmap puede llegar a ser sencillo, aunque tiene un despliegue de opciones que, a buen seguro, mucha gente encontrará interesante.

Una ejecución de este programa puede dar lugar a un resultado como este:

```
Starting Nmap 4.53 ( http://insecure.org ) at 20080603 12:05 CEST Interesting ports on 192.168.1.1: Not shown: 1711 closed ports PORT STATE SERVICE 21/tcp open ftp 23/tcp open telnet 80/tcp open http MAC Address: 00:02:CF:81:6F:89 (ZyGate Communications)
```

Nessus, en cambio, nos ofrecerá una herramienta cliente/servidor que utilizará una base de datos con las vulnerabilidades que estadísticamente han podido ocasionar “desastres” y nos avisa mediante este escaneo.

La interfaz, además, es bastante más amigable y nos mostrará unas estadísticas de los procesos ejecutados.

Escaneo de vulnerabilidades web



Más en consonancia con el objetivo de este artículo, están los escaneos de vulnerabilidades propiamente web. Estos escaneos se pueden basar en varias premisas, empleando sistemas de conocimiento, funciones heurísticas e incluso técnicas *fuzz*, que veremos más adelante. Una buena combinación de estos elementos puede darnos muchas pistas a la hora de proteger nuestro site y llegar donde nosotros no alcanzamos. Empecemos por los escaneadores automáticos más empleados y populares.

Acunetix

Acunetix, que goza de una versión **Free Edition (sólo para HTML Injection)**, pero con una gran variante de sistemas de inyección, una base de datos amplia y una interfaz muy amigable. Los procesos por los que puede “atacarse” pueden ser varios y los perfiles de ataque – si se tiene la versión de pago – de los más variopintos, muchos de ellos ya los hemos visto aquí.

SSS (Shadow Security Scanner)

Similar al anterior en cuanto a **sistema web, quizá no tan completo, pero que ofrece también el sondeo de otros protocolos como FTP, NetBios**, módulos de Apache del que se tengan constancia que hay vulnerabilidades.

Técnicas Fuzz

Se llama *fuzzing* a las **diferentes técnicas de testeo de aplicativos que generan datos secuenciales y aleatorios para obtener así las vulnerabilidades de la victima**. Este sistema puede ser muy potente, pues combina la aleatoriedad en los ataques con ataques basados en formatos heurísticos. Una lista de estos potentes escaneadores de vulnerabilidades pueden encontrarse en:

www.infosecinstitute.com/blog/2005/12/fuzzers-ultimate-list.html

Un ejemplo lo podemos tener ejecutando el WebFuzzer, con licencia GPL, escrito en C:

[/gunzip.altervista.org/g.php?f=projects#webfuzzer](http://gunzip.altervista.org/g.php?f=projects#webfuzzer)

PHP IDS



PHP-IDS es un sistema basado en PHP que actúa como IDS (Intrusion Detect System) y que se aplica a todos nuestros archivos buscando algún tipo de inyección o vulnerabilidad. Puede detectar desde XSS, SQL Injection, RFI y ataques LDAP Injection y tiene incluso hasta módulos especializados para distintos tipos de CMS.

Módulos Apache

De entre ellos, existen muchos que nos pueden ayudar a nuestro cometido, aunque nos centraremos en los siguientes:

Mod_rewrite Famoso sobre todo para el uso de **URL-Friendly**, pues reescribe la entrada transformándola en otras “Human readability”. Personalmente recomiendo el uso de **mod_security**, debido a que **mod_rewrite** tiene lógicas limitaciones al no ser un módulo diseñado a tal efecto.

Conclusión

No es un caso trivial tener que proteger un site web, tanto si ya está hecho como si lo tenemos que desarrollar de nuevo. La única forma de obstaculizar el ejercicio de estos atacantes será conocer cuáles son sus técnicas, mantenerse actualizado regularmente de las vulnerabilidades de nuestro entorno (**Sistema Operativo, Lenguaje, base de datos y módulos y librerías asociados**), **en caso de ser un programa conocido (como un WordPress, Joomla!, PostNuke)** mantenerse alerta a los *bugs* que, altruistamente, algunos atacantes publican en webs.

Además, con un **sistema IDS** que nos pueda ir comunicando qué pasa con nuestros *logs*, la evolución de estos mismos y la constante evaluación de las vulnerabilidades de nuestro sistema, junto con un escaneo automático, **técnicas fuzz** y una programación sólida, y algún módulo destinado a la seguridad harán de nuestro servidor web una fortaleza (casi) inexpugnable.

Este artículo es obra de *Brian Hernández*
Fue publicado por primera vez en 13/06/2013
Disponible online en <http://desarrolloweb.com/articulos/seguridad-proyectos-php.html>

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 16/08/2007
Disponible online en <http://desarrolloweb.com/articulos/problema-error-404-php.html>

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.

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

[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

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.

Este artículo es obra de *Miguel Angel Alvarez*
Fue publicado por primera vez en 22/09/2009
Disponible online en <http://desarrolloweb.com/articulos/librerias-json-php.html>