

PHP

En este capítulo encontraremos una serie de notas referentes al lenguaje de programación PHP: cómo adquirirlo, sus versiones, sus principales características y demás temas que servirán de base para los capítulos subsiguientes.

Obtención	16
Licencia de uso	17
Versiones	18
PHP 3	19
PHP 4	19
PHP 5	20
¿Qué versión utilizar?	22
Extensiones en PHP	23
Bases de datos	25
Ventajas de trabajar con extensiones	26
Bibliotecas incorporadas	26
Facilidad de aprendizaje	28
Portabilidad	28
Resumen	29
Actividades	30

OBTENCIÓN

PHP, acrónimo de **Hypertext Preprocessor**, es un lenguaje de programación que se utiliza en la mayoría de los casos para el desarrollo de sitios web, pero que para muchos es un lenguaje de propósito general y el uso que se le dé dependerá en gran parte de las necesidades que posea el programador.

Entre las características que hacen de PHP un lenguaje popular y muy poderoso para desarrollar aplicaciones, podemos citar las siguientes:

- Programación de páginas dinámicas en servidores.
- Programación de aplicaciones de escritorio con GTK (*PHPGTK*).
- Soporte para trabajar con múltiples bases de datos.
- Soporte para múltiples plataformas.
- Soporte para múltiples servidores.
- Facilidad de aprendizaje.
- Portabilidad de código entre diferentes plataformas.
- Total libertad para distribuir las aplicaciones.

Para obtener una copia de PHP, deberemos ingresar a su sitio web, **www.php.net/downloads/**, y seleccionar la opción que coincida con nuestro sistema operativo.

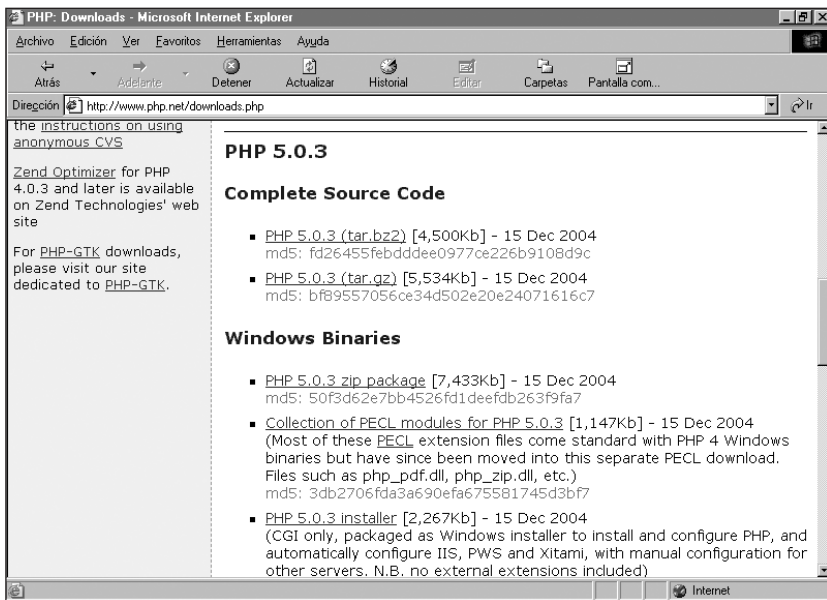


Figura 1. Área de descargas en el sitio web de PHP.

Según Netcraft (**www.netcraft.com**), compañía que se dedica entre otras cosas a brindar estadísticas acerca del uso de tecnologías en Internet desde 1995, la utiliza-

ción de PHP en servidores viene creciendo en forma sostenida y se hace cada vez más popular, como se ve en la figura.

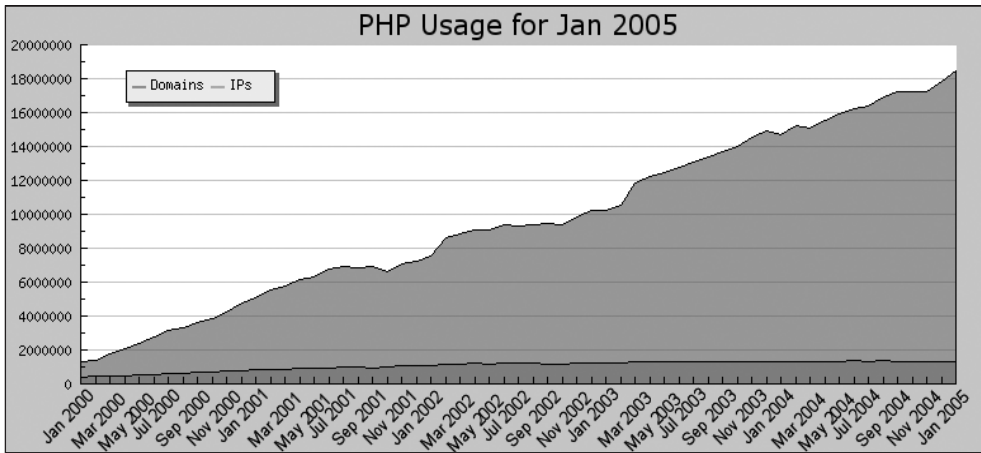


Figura 2. Uso de PHP en servidores a través del tiempo. (Fuente: <http://news.netcraft.com>).

LICENCIA DE USO

Se mencionó con anterioridad que PHP es un lenguaje “libre”. Este término se refiere al tipo de licencia que tiene, y consiste, básicamente, en tres puntos o “libertades”:

- La **primera** libertad es la de usar el programa (PHP).
- La **segunda** es la de poder modificar el programa accediendo a su código fuente.
- La **tercera** es la de distribuir el programa modificado o no.

La licencia de PHP está disponible en www.php.net/license o, también, viene junto con el programa en el archivo **license.txt**.

La redistribución, modificación y uso de PHP están permitidos bajo las siguientes normas (licencia versión 3.0):



MI NOMBRE ES MI NOMBRE

PHP es en realidad un acrónimo recursivo de **PHP Hypertext Pages**, ya que utiliza su propio acrónimo como parte de su acrónimo. También ocurre esto con **GNU**, acrónimo recursivo de **GNU'S Not Unix**. En el caso de PHP se eligió la primera letra P (que podría haber sido cualquier otra) porque en algún momento (en los inicios del lenguaje) éste se llamó **Personal Home Page Tools**.

- La redistribución del código fuente de PHP debe estar siempre acompañada de la licencia y copyright de PHP.
- No puede usar el nombre **PHP** para promocionar sus productos, a menos que tenga permiso por escrito del **PHP Group**.

No hay una empresa comercial detrás de PHP; las continuas mejoras y avances dentro del lenguaje resultan de una gran comunidad de desarrolladores que contribuyen, sin obtener réditos comerciales, con: código fuente, soporte a otros usuarios a través de listas de correo, revisión del programa en busca de errores, notificación de fallas de seguridad, y más.

Sobre esta base se sostiene una licencia que, justamente, asegura la libertad del lenguaje y no permite bajo concepto alguno que alguien obtenga beneficios comerciales de PHP y sea el dueño del lenguaje: éste es el espíritu de la licencia.

Cuando se desarrolla una aplicación y se la vende a terceros el importe que se cobra no es el lenguaje de programación sino la solución a un problema, el tiempo invertido en el desarrollo, el soporte, u otro particular.

VERSIONES

Esta sección no pretende repasar la historia de PHP, sino que, por el contrario, tiene como objetivo realizar una breve reseña de las características principales y los cambios que sufrió el lenguaje desde la versión **3** hasta la **5** inclusive.

PHP tuvo versiones anteriores a la 3, pero elegimos ésta como punto de partida porque es la versión más antigua que puede verse hoy en día en algunos sitios.

Esto sucede, mayormente, porque los sitios en cuestión –que fueron desarrollados en origen cuando esta versión era la última– cumplen con un objetivo preciso y no han necesitado actualización a versiones posteriores.

Hoy, la mayoría de las aplicaciones están programadas con la versión **4**, aunque están migrando de forma constante a PHP **5**.



EL PRINCIPIO

PHP nació originalmente como un contador de visitas al sitio web que contenía el currículum de su creador Rasmus Lerdorf en 1995. Estaba programado en PERL. Lerdorf realizó algunas mejoras y agregó nuevas funcionalidades para luego liberar el código fuente al público. En 1997 Andi Gutmans, Zeev Suraski y Lerdorf rescribieron el código y anunciaron la versión 3 de PHP.

PHP 3

PHP versión 3 fue creado en 1997 y se lo considera una continuación de una versión anterior de PHP llamada **PHP/FI 2.0**, aunque llamarla “continuación” es relativo porque el código se rescribió completamente, y sólo se mantuvieron su forma de trabajar y algunas funciones implementadas: la idea era mantener a los usuarios de **PHP/FI** y seguir trabajando en conjunto con ellos.

Ya en esta versión, PHP tenía características que perduraron en el tiempo y aún hoy son un punto fuerte del lenguaje: soporte para una gran cantidad de bases de datos, interacción con protocolos de red y uso de extensiones.

En cuanto a la orientación a objetos (*POO*), la versión 3 le daba soporte a medias, sin implementar todas las posibilidades de este paradigma.

Se vislumbraba ya por estos tiempos a una gran cantidad de personas (decenas de millares de usuarios y cientos de miles de sitios web) que se encontraban interesadas no únicamente en utilizar sino, también, en colaborar con el lenguaje. PHP 3.0 se lanzó de manera oficial en el mes de junio de 1998.

PHP 4

Tomando como punto de referencia la versión anterior, el núcleo (parte del programa que se encarga de administrar los procesos) de PHP fue rescrito para la versión 4. Esto se dio porque al ser cada vez más popular, las aplicaciones existentes en el mercado fueron haciéndose más complejas y requerían mayor velocidad en la ejecución que la que podía ofrecer PHP versión 3.

El nuevo núcleo se denominó **Motor Zend** (en referencia a los nombres de sus principales desarrolladores, **Zeev Zuraski** y **Andi Gutmans**).

Otras mejoras importantes son: el soporte para la mayoría de los servidores web, las funciones para el manejo de sesiones HTTP, los buffer de salida y la inclusión de gran cantidad de funciones de propósitos diversos.

La versión 4 llegó a estar instalada en más del 20% de los dominios en Internet. En cuanto a la POO, el soporte que PHP ofrecía, prácticamente, no se llegó a modificar con relación a la versión 3. Lo que sí se modificó fue su uso, ya que se volvió habitual para gran cantidad de usuarios, generalmente, en aplicaciones de gran tamaño. Este



AVANZAR EN DISTINTAS DIRECCIONES A LA VEZ

Dentro de la amplia oferta de lenguajes de programación –destinados al desarrollo de aplicaciones tanto web como a medida–, existen actualmente algunos que tienen una sintaxis muy similar a la de PHP. Ésta es, quizás, otra de las ventajas de este lenguaje: aprender a programar en él, da la posibilidad de hacerlo luego en otros.

requerimiento casi explícito por parte de los usuarios tendría su respuesta en la versión 5. Un caso similar se dio con XML: PHP daba soporte para manejar esta clase de archivos pero era bastante limitado en cuanto a las opciones que ofrecía. Un punto importante es que las extensiones escritas para PHP 3 no trabajan con PHP 4 (aunque es posible portar extensiones a PHP 4 si se tiene acceso a las fuentes originales). PHP 4 se introdujo por primera vez en el mercado en el año 1999 y la versión oficial fue lanzada en el mes de mayo de 2000.

PHP 5

Los cambios que experimenta PHP en esta versión son verdaderamente importantes, y se podría decir que revolucionaron el lenguaje.

La primera versión de PHP 5 fue liberada en junio de 2003. Entre las nuevas características que incluía, podemos destacar:

- Mejoras sobre el motor Zend (**Zend Engine II**).
- En cuanto a la programación orientada a objetos, PHP ahora ofrece notables mejoras que lo hacen una alternativa totalmente competente en este aspecto en comparación a otros lenguajes con historia en este campo.
- Mejoras en el soporte de XML (el código fue rescrito en comparación a la versión anterior).
- Manejo de excepciones (similar a como lo hace Java).
- Inclusión de soporte para SQLite.

SQLite

SQLite es una librería escrita en lenguaje C que implementa un motor de base de datos accesible por varios lenguajes (PHP, PYTHON, C, etc.).

SQLite no permite múltiples usuarios accediendo a la vez en modo escritura a la base de datos, debido a que el mecanismo de bloqueo que utiliza consiste en bloquear toda la base de datos. Por eso esta librería está especialmente indicada cuando se requiera de una gran rapidez en las consultas y nos baste que un único usuario pueda realizar modificaciones (SQLite es entre dos y tres veces más rápido que bases de datos como MySQL o PostgreSQL, ya que no es un servidor sino una base de datos de escritorio, por lo que el acceso a los datos es más directo, entre otras razones). Brinda soporte para un máximo de 2 terabytes de datos.

SQLite es software libre y podremos obtener más información en su sitio oficial www.hwaci.com/sw/sqlite/.

Pero a pesar de estos importantes avances, quizás nos interese, particularmente, la inclusión de **MySQLi** (con **i** de “**improved**”, *mejorado*), una extensión que permite acceder a las funcionalidades provistas por MySQL a partir de la versión 4.1.2. Veremos dos características: las consultas preparadas y la seguridad en las conexiones.

Consultas preparadas

Cuando ejecutamos una instrucción SQL lo que sucede es, en resumidas cuentas, que se crea la instrucción en PHP, se la envía al servidor de bases de datos (MySQL), éste controla que la instrucción sea válida y devuelve un resultado (puede ser un conjunto de registros, por ejemplo). Luego PHP trabaja con esos datos.

Si ejecutamos una instrucción **N** veces (por ejemplo, insertar 100 registros), hay que enviar **N** instrucciones a MySQL.

La posibilidad que da MySQL desde la versión 4.1.2 es la de que, en vez de recibir **N** instrucciones, permite aceptar consultas preparadas: PHP le envía un **molde** de la instrucción, MySQL lo valida y lo guarda, luego PHP no tiene que enviarle la instrucción completa sino que puede enviarle sólo los datos que podrían cambiar de una instrucción a otra. Por ejemplo, trabajando sin consultas preparadas se haría lo siguiente:

```
INSERT INTO tabla1 VALUES (1, "argentina", 100.000);
INSERT INTO tabla1 VALUES (2, "brasil", 500.000);
INSERT INTO tabla1 VALUES (3, "uruguay", 300.000);
INSERT INTO tabla1 VALUES (4, "paraguay", 200.000);
INSERT INTO tabla1 VALUES (5, "chile", 600.000);
```

En este caso, MySQL hace 5 veces un trabajo similar, sin contar el problema del tráfico en la red.

En cambio, con consultas preparadas sería:

```
INSERT INTO tabla1 VALUES (?, ?, ?);
```

Así, se envía y se valida una sola vez.

```
(1, "argentina", 100.000);
(2, "brasil", 500.000);
(3, "uruguay", 300.000);
(4, "paraguay", 200.000);
(5, "chile", 600.000);
```

Vemos que se envían menos datos.

Lo anterior no está codificado en PHP. Simplemente sirve para ilustrar el concepto de las consultas preparadas (*Prepared Statements* en inglés).

Seguridad en las conexiones

Utilizando las funciones provistas por PHP para acceder a bases de datos MySQL, existía la posibilidad de tener una conexión por defecto: se establecía una conexión y si en las siguientes instrucciones SQL no se la indicaba expresamente, se asumía la última abierta. Esto puede traer ciertos problemas relacionados a la seguridad, por eso MySQLi requiere que se especifique explícitamente en cada **script** PHP la conexión que se usa.

Se ha probado que esta extensión ofrece mayor velocidad (en algunas operaciones es hasta cuarenta veces más rápida que la extensión MySQL anterior).

Se está trabajando para que la migración desde la extensión de MySQL a MySQLi sea lo menos trabajosa posible. Esto se logra, entre otras cosas, manteniendo las funciones –y sus nombres– lo más parecidas que se pueda en ambas extensiones. Si vemos códigos programados con ambas extensiones, no notaremos grandes diferencias en general. Podemos encontrar una reseña acerca de cómo instalar la extensión MySQLi en www.php.net/mysql/.

La primera versión considerada estable de PHP 5 se liberó en julio de 2004.



Figura 3. Se puede encontrar información y características de la última versión de PHP en su sitio oficial.

¿Qué versión utilizar?

Si la aplicación ya está funcionando y cumple con sus propósitos, evidentemente no hay razón para migrar a otra versión de PHP.

Ahora bien, si queremos desarrollar un sitio desde cero, lo que conviene es utilizar la versión más reciente del lenguaje: quizás no utilicemos todas las mejoras ni todas sus características; pero si quisiéramos hacerlo no tendríamos que reescribir el código, ni actualizar a la versión más reciente, ni entrar en temas en ocasiones engorrosos como es la compatibilidad del lenguaje entre dos versiones distintas del mismo. Aunque a veces se diga que esta decisión depende de los requerimientos del desarrollador, PHP versión 5 es un tema aparte porque, como se dijo, ha sido mejorado y no de forma cosmética: el soporte mejorado para la programación orientada a objetos es algo que ya es muy popular en otros lenguajes y PHP no es la excepción. Gran cantidad de código está siendo escrito bajo este paradigma y mientras más rápido se acostumbre el programador PHP, más en carrera estará a la hora de actualizar sus conocimientos e incluso al momento de buscar empleo.

Si tenemos instalada una versión de PHP y no sabemos cuál es, se puede recurrir a la función **PHPVERSION()**:

```
<?php
// EJEMPLO DE PHPVERSION()
echo "Version de PHP: ".PHPversion();
?>
```

EXTENSIONES EN PHP

Cuando programamos en un lenguaje –no sólo en PHP–, normalmente nos valemos de funciones o procedimientos, ya sea para resolver problemas o modular el código para hacerlo más legible y reutilizarlo sin escribir lo mismo más de una vez. Las extensiones no son ni más ni menos que conjuntos de funciones que tenemos disponibles para programar pero, para ser más precisos, podríamos dividir las funciones en dos grupos: las funciones que vienen incorporadas con el lenguaje



EXTENSIÓN DE LA DISTRIBUCIÓN

Al momento de descargar la última versión de PHP desde su sitio oficial, encontrará archivos con distintas extensiones (.zip, .tar.bz, tgz). Debe saber que todos contienen la misma información, y la elección de descargar uno u otro dependerá de si tiene o no un programa descompresor que pueda trabajar con dichas extensiones.

(llamadas **built in**) y las que están en las bibliotecas añadidas, que se tienen que instalar en el sistema de forma específica.

Las extensiones –o bibliotecas– componen el segundo grupo. Se podría decir que para utilizar ciertas funciones hay que **extender** el lenguaje.

Una vez que se instalan y se habilitan esas bibliotecas (ya veremos cómo hacerlo), el comportamiento de las funciones componentes dentro del código de nuestros programas es idéntico al de cualquier función o procedimiento, o sea que la programación se vuelve independiente y transparente al origen de las funciones.

Las extensiones en PHP pueden agruparse por funcionalidad, es decir que podríamos encontrarnos con una extensión para manipular cadenas de caracteres, otra para acceder a bases de datos, otra para trabajar con archivos XLS, y demás.

Para instalar estas extensiones, tenemos que seguir una serie de pasos: el primero de ellos es abrir el archivo **php.ini** (si estamos trabajando con PHP versión 3 el archivo, probablemente, se llame **php3.ini**; desde la versión 4 en adelante se ha convenido en llamarlo simplemente **php.ini**), buscar la línea que comienza con **extension_dir** y reemplazarla por **extension_dir = "c:\PHP\extensions"**, en caso de que **C:\PHP\extensions** sea en donde tenemos almacenadas las extensiones que vienen incluidas con PHP. Para saber en qué lugar se encuentran las extensiones, debemos buscar dentro del directorio en donde instalamos PHP una carpeta llamada **extensions** (o **ext**) y copiar la ruta completa hasta allí a la línea **extension_dir** del **php.ini**, tal como se mostró anteriormente.

Guardar las extensiones en un solo lugar implica el beneficio de no tener que indicar la ruta completa a la extensión cada vez que queramos usarla. Según estemos trabajando sobre sistemas **Linux/Unix** o **Windows** habrá diferencias tales como:

- Linux usa la barra / para acceder a subdirectorios (el directorio de extensiones podría ser **/usr/lib/PHP3**), mientras que Windows usa la barra \.
- Las extensiones en Linux son archivos **.SO**, en Windows son **.DLL**.

Ahora sólo nos queda habilitar las extensiones que vayamos a usar. Lo único que debemos hacer es buscar dentro del archivo **php.ini** una lista del tipo (con archivos terminados en **.SO** o **.DLL** según corresponda):

CÓDIGO DE BARRAS

Una de las equivocaciones más frecuentes a la hora de publicar nuestros archivos PHP en un servidor con un sistema operativo diferente del nuestro, es la de escribir las rutas a archivos o directorios de forma incorrecta. Recuerde que los sistemas Unix compatibles usan / para acceder a subdirectorios y los sistemas Windows usan \.

```

;extension=pgsql.so
;extension=mysql.so
;extension=gd.so
;extension=imap.so
;extension=ldap.do
;extension=xml.so
;....

```

Y quitar el punto y coma inicial para habilitar las bibliotecas que queramos usar. Para que los cambios tengan efecto habrá que reiniciar el servidor web. Se pueden habilitar múltiples extensiones, incluso no existe ningún inconveniente en, por ejemplo, utilizar dos bibliotecas con el objetivo de acceder a diferentes tipos de bases de datos al mismo tiempo.

Bases de datos

PHP tiene extensiones para soportar, entre otras, las siguientes bases de datos:

- DBase
- Informix
- Interbase/Firebird
- MS SQL Server
- Mysql
- msql
- Oracle
- Postgre SQL
- Sybase

Incluso con **ODBC** (*Open Data Base Connectivity, Conectividad Abierta de Bases de Datos*) se puede acceder a casi cualquier base de datos existente en el mercado: ODBC brinda un conjunto de comandos que son traducidos a instrucciones específicas de una base de datos en particular a través de drivers provistos por éstas.

Es claro que utilizar funciones nativas da más réditos, en cuanto a la velocidad de respuesta, comparado con trabajar con algún mediador tipo ODBC.



INSTALACIÓN

Hay programas que nos ofrecen descargar PHP, MySQL y Apache en un solo archivo, e instalarlos a través de un programa que nos guía y nos pregunta dónde queremos instalar cada elemento, qué tipo de instalación preferimos, etc.. Se recomienda instalar estas aplicaciones “a mano”, ya que así aprenderemos a configurarlas. Claro que la decisión final queda a criterio del lector.

Por otro lado cuando se emplean bases de datos con una gran cantidad de prestaciones como **Oracle** y se utiliza **ODBC**, **OLE**, **ADO**, etcétera, se pierde gran parte del poder, puesto que hay funciones propias de la base de datos que no se pueden utilizar con un mediador genérico como éstos, por no estar implementadas. Por lo tanto si se quiere utilizar todo el poder de la base de datos, es preferible acceder con funciones nativas para ésta, como las que ofrece PHP en sus extensiones. Por ejemplo, si en nuestro proyecto sabemos que vamos acceder sólo a la base de datos **Postgre SQL**, no hay necesidad de acceder con ODBC ya que PHP nos provee de una extensión (**php_pgsqll.dll**) para hacerlo de forma nativa.

Ventajas de trabajar con extensiones

Sólo cargamos las bibliotecas cuando las usamos: PHP ya tiene demasiadas funciones incorporadas y sería poco recomendable iniciar el motor para soportar cientos de funciones de las cuales probablemente necesitemos sólo algunas. Además, al añadir una biblioteca no hace falta reinstalar PHP, sólo habilitar desde el archivo `php.ini` lo que necesitamos: esto significa modularidad.

Bibliotecas incorporadas

PHP incorpora sin necesidad de ningún tipo de instalación ni habilitación extras las siguientes funciones:

- Para manejo de matrices.
- Funciones matemáticas.
- BCMath (Desde PHP 4.0.4. Más funciones matemáticas).
- Para manejo de Clases/Objetos.
- Para manejo de variables de tipo de carácter.
- Para tratamiento de Fecha y Hora.
- Para acceso directo a Entrada/Salida.
- Funciones de directorio.
- Funciones de Gestión de Errores y Registros.



MIRAR Y APRENDER

Se logra un mayor entendimiento del lenguaje cuando se lee código escrito por otras personas: un mismo problema puede resolverse de muchas maneras, y no quedarse sólo con un punto de vista ayuda a abrir nuestra mente e incorporar nuevas formas de encarar la escritura de un código.

- Funciones de Sistema de Archivos.
- Para utilizar el protocolo FTP.
- Para utilizar el protocolo HTTP.
- Funciones de correo.
- Funciones de Red.
- Funciones de Control de Salida.
- Para ejecución de Programas.
- Funciones para el manejo de sesiones.
- Funciones de secuencia.
- Funciones de cadenas.
- Funciones URL.
- Para manejo de Variables.

Para poder tener acceso a las demás bibliotecas, tendremos que activarlas a través del archivo **php.ini**, o bien incorporarlas al momento de compilar PHP e instalar las bibliotecas en forma separada (únicamente para aquellos sistemas operativos en los que para instalar PHP haya que compilarlo).

Para ver qué bibliotecas tenemos activas en nuestro sistema, podemos utilizar la función **PHPinfo()** de la siguiente manera:

```
<?PHP

// funcion PHPinfo
echo PHPinfo();

?>
```

Normalmente cuando no recordamos cómo instalar o habilitar una biblioteca en particular en nuestro sistema, bastará con buscar la referencia en el manual de PHP, en www.php.net/manual/.

LAMP

En revistas especializadas, libros y/o artículos en Internet encontrará referencias a los sistemas LAMP. Esta sigla se refiere a Linux, Apache, MySQL, y alguno de los lenguajes Perl/PHP/Python. Los sistemas tipo LAMP son muy utilizados en la actualidad, entre otras razones, debido a la estabilidad, la potencia y el bajo costo de desarrollo de las herramientas que los componen.

FACILIDAD DE APRENDIZAJE

Esto evidentemente no equivale a hacer un juicio de valor sobre otros lenguajes, simplemente se trata de una cualidad de PHP.

PHP se caracteriza por ser un lenguaje cuyo aprendizaje se vuelve sencillo incluso para aquéllos que nunca han trabajado con ningún otro lenguaje de programación, pero está claro que tener conocimientos previos ayuda a entender más rápidamente qué se está haciendo y cómo se está haciendo.

Un punto importante es que la sintaxis de PHP deriva o es similar a la del **lenguaje C**, que es realmente muy popular: el que tenga conocimientos en este lenguaje se acercará con más facilidad a PHP.

La meta de este lenguaje es permitir escribir a los creadores de sitios web páginas dinámicas de una manera rápida y fácil, pero si se quiere ir más allá y llegar a desarrollar aplicaciones complejas necesitaremos avanzar dentro del lenguaje: quizás PHP sea fácil de aprender pero este aprendizaje debe ser constante.

PORTABILIDAD

PHP es un lenguaje multiplataforma, lo que significa que está preparado para trabajar sobre distintos sistemas operativos. Más adelante en este mismo capítulo podremos ver un listado de los sistemas operativos soportados.

Pero la portabilidad está también en que no es necesario realizar grandes modificaciones al código fuente de una aplicación escrita en PHP al momento de trasladarla de una plataforma a otra: si lo deseamos podemos desarrollar nuestra aplicación en Windows o MAC, pero luego subir el mismo código a un servidor que esté corriendo Linux, por ejemplo.

La portabilidad de PHP es, sin duda, un punto fuerte frente a lenguajes como ASP, que necesitan de componentes adicionales para correr en algunas plataformas. PHP corre en una gran cantidad de sistemas operativos y sin necesidad de un componente adicional que debamos comprar.

PHP está disponible para los siguientes sistemas operativos:

- Unix/HP-UX
- Unix/Linux
- Unix/Mac OS X
- Unix/OpenBSD
- Unix/Solaris
- Unix
- Windows (todas las versiones, salvo PHP5 que no corre bajo Windows 95)
- MAC

Actualmente, PHP se puede ejecutar bajo los servidores web Apache (incluso en la versión 2.0), IIS (*Internet Information Server*), PWS (*Personal Web Server*), AOLServer, Roxen, OmniHTTPd, O'Reilly Website Pro, Sambar, Xitami, Caudium, Netscape Enterprise Server, THTTPD, y otros.

Las posibles incompatibilidades entre plataformas son las siguientes:

- Ruta a archivos/directorios.
- Bibliotecas que sólo funcionan en algunos sistemas, por ejemplo:
 - Funciones W32api (sólo para plataformas Windows de 32 bits).
 - Funciones para el manejo de Impresoras (sólo en Windows 9x, ME, NT4 y 2000).
 - Las funciones COM para Windows.
 - Funciones de acceso directo a E/S (no disponibles para sistemas Windows).
 - Funciones GMP (funciones que permiten trabajar con enteros de longitud variable, no disponibles para sistemas Windows).
 - Funciones para el control de procesos (no disponibles para sistemas Windows).
 - Funciones FAM (notifican cambios en archivos y directorios, no disponibles para sistemas Windows).
 - Funciones POSIX (no disponibles para sistemas Windows).
 - Funciones para Ncurses (no disponibles para sistemas Windows).

Incluso existen bibliotecas que ofrecen compatibilidad con Windows NT/2000/XP pero no con las versiones de Windows 9.x.

Normalmente cuando no recordamos si una biblioteca es o no compatible con nuestro sistema, basta con buscar la referencia en el manual de PHP, www.php.net/manual/ y verificar si hay algún requerimiento.

RESUMEN

En este capítulo hemos visto una reseña de las principales virtudes del lenguaje de programación PHP: qué hace, para qué sirve, en qué se usa mayormente, qué versiones existen. También vimos cuáles son los requisitos para instalarlo y cómo incorporar extensiones para hacer nuestro trabajo más fácil. En definitiva, un punto de partida para conocer un lenguaje que está dando mucho que hablar y que seguirá haciéndolo.



TEST DE AUTOEVALUACIÓN

- 1 ¿Cuánto dinero hay que pagar para poder usar PHP?

- 2 ¿Qué es una extensión?

- 3 ¿Cuál es la ventaja principal de trabajar con extensiones?

- 4 ¿En qué directorio de su sistema se encuentran las extensiones de PHP?

- 5 ¿En qué versión de PHP se incorporó la extensión MySQLi? ¿Para qué sirve?

- 6 ¿Podría nombrar cuatro razones por las cuales usaría PHP para programar sus aplicaciones?

- 7 ¿PHP sólo sirve para programar desarrollos web?

- 8 Nombre tres de las nuevas características que vienen con PHP 5.

- 9 Brevemente, ¿qué es SQLite?

- 10 ¿Desde qué sitio se puede descargar la última versión de PHP?

- 11 ¿PHP puede trabajar con múltiples servidores web o sólo con Apache?

- 12 ¿PHP puede trabajar con múltiples bases de datos o sólo con MySQL?

EJERCICIOS PRÁCTICOS

- ✓ ¿Cuál es su versión de PHP? Responda utilizando la función de PHP que devuelve tal información.

- ✓ ¿Qué extensiones tiene habilitadas en su sistema?
Responda inspeccionando el archivo php.ini o bien a través de la función phpinfo.
