PHP

UT1: INTRODUCCIÓN

1. Introducción a PHP y MySQL
2. Breve historia de PHP
3. Breve historia de MySQL
4. Servidores web de desarrollo y explotación
5. La aplicaciones ¡Hola mundo! con PHP
6. Necesito ayuda: php.net

www.php.net/manual/es/

1. Introducción a PHP y MySQL

PHP es un lenguaje de programación que se ejecuta en servidores web y permite crear páginas HTML de forma dinámica. Las siglas PHP provienen de **Personal Home Page**, que podemos traducir como Procesador personal de páginas web o de hipertexto.

Las páginas web se almacenan en archivos de texto, es decir, archivos que contienen caracteres legibles.

Nota: En última instancia, cualquier información que maneje un ordenador debe expresarse como una secuencia de bits (ceros y unos). Los archivos de texto en los que se almacenan las páginas web no son una excepción, y pueden utilizarse distintos estándares para codificar sus caracteres legibles en combinaciones de ceros y unos; los más populares son los derivados del ANSI y Unicode, dentro de los que destacan el ISO 8859-1 y el utf-8, respectivamente. Los derivados del ANSI utilizan un único byte para codificar cada carácter, de modo que sólo pueden representar 256 caracteres distintos. Sin embargo, los derivados del Unicode utilizan varios bytes para codificar cada carácter, de modo que son capaces de representar virtualmente cualquier carácter de cualquier lengua viva o muerta, incluidos símbolos matemáticos y musicales. Por ejemplo, la letra á en ISO 8859-1 se codifica como 11100001 (0xE1) y en utf-8 como 1100001110100001 (0xC3 A1); sin embargo, la letra a en ambos sistemas se codifica como 1100001 (0x61). Como programadores de PHP y MySQL, y especialmente por la existencia en nuestro idioma de caracteres especiales como las letras acentuadas o la eñe, nos encontraremos con situaciones en las que necesitemos realizar conversiones de unos estándares de codificación a otros; por ejemplo, si el contenido de una base de datos está codificado en ISO 8859-1 pero queremos mostrarlo en una página web codificada en utf-8.

Cuando escribimos una dirección en nuestro navegador web estamos solicitando que otro ordenador de Internet nos envíe un archivo con el contenido de la página que queremos ver. Nuestro ordenador está actuando en este caso como *cliente* del ordenador *servidor* en el que se encuentra almacenada la página, o más exactamente, nuestro navegador web está actuando como cliente de la aplicación de servidor web disponible en el otro ordenador (tradicionalmente se manejan los términos ordenador servidor y ordenador cliente, pero actualmente es más adecuado referirse a aplicaciones servidor y aplicaciones clientes, pues en un mismo ordenador pueden estar funcionando simultáneamente aplicaciones de uno y otro tipo).

Ese archivo debe estar codificado conforme a las normas del estándar HTML (XHTML), que es el que son capaces de interpretar los navegadores web.

A continuación se muestra un ejemplo básico de archivo de página web en formato HTML:

```
001  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
003          <head>
004                  <title>T&iacute;tulo de la p&aacute;gina</title>
005                  <meta http-equiv="content-type" content="text/html; charset=utf-8">
006          </head>
007          <body>
008                  <p>Ha accedido a las 9:20</p>
009          </body>
010  </html>
```

Este archivo es de tipo estático, pues siempre que accedamos a él nos indicará que son las 9:20,

independientemente de cuál sea la hora real.

Sin embargo, gracias al lenguaje PHP podremos insertar en los archivos HTML instrucciones que serán interpretadas por el servidor antes de enviar el archivo al cliente. Por ejemplo, el siguiente archivo mostraría la hora *del servidor* actualizada cada vez que se accediera a él.

```
001  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
003          <head>
004                  <title>T&iacute;tulo de la p&aacute;gina</title>
005                  <meta http-equiv="content-type" content="text/html; charset=utf-8">
006          </head>
007          <body>
008                  <p>Ha accedido a las <?php echo date('G:i');?></p>
009          </body>
010  </html>
```

Obsérvese que se muestra la hora del servidor pues todo lo escrito entre los elementos `<?php` y `?>` es interpretado por el servidor antes de enviar el archivo al cliente. Por ejemplo, si fueran las 9:20 en el ordenador servidor, el archivo que recibiría el cliente sería exactamente el mismo que en el ejemplo de la página estática anterior. Por este motivo PHP es un lenguaje interpretado del lado del servidor, es decir, *server-side scripting*.

Esto contrasta con otros lenguajes de programación interpretados del lado del cliente (*client-side scripting*) como JavaScript. Por ejemplo, en el siguiente archivo todo el código insertado entre los elementos <script> sería interpretado por la aplicación cliente cada vez que se accediese a la página, de modo que mostraría en cada acceso la hora configurada en el ordenador cliente.

```
001  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
     "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
002  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es-ES" lang="es-ES">
003          <head>
004                  <title>T&iacute;tulo de la p&aacute;gina</title>
005                  <meta http-equiv="content-type" content="text/html; charset=utf-8">
006          </head>
007          <body>
008                  <p>Ha accedido a las <script type='text/javascript'>var hora=new
     Date();document.write(hora.getHours()+':'+hora.getMinutes());</script></p>
009          </body>
010  </html>
```

Consecuentemente, lo primero que debemos tener siempre presente sobre PHP es que es un lenguaje que se interpreta en el lado del servidor y que, por tanto, sólo puede disponer (conocer o acceder) de los recursos del ordenador servidor; por ejemplo, una aplicación PHP no podrá acceder a los archivos presentes en el ordenador cliente.

¿En qué escenarios es conveniente utilizar PHP si existen otras alternativas del lado del cliente que descargarían de trabajo al servidor? Justamente siempre que necesitemos manejar información sensible que no deseemos que esté al alcance del cliente y, fundamentalmente, por los dos motivos siguientes:

2. **Seguridad**. Por ejemplo, imaginemos un formulario de acceso por credenciales; si utilizásemos un lenguaje interpretado del lado del cliente deberíamos enviarle las credenciales de todos los usuarios registrados para que pudiese averiguar si las credenciales

escritas por el usuario coinciden con las de algún usuario registrado. Obviamente esto es un despropósito, pues un usuario malintencionado podría acceder a esas credenciales y hacerse pasar por un usuario legítimo.

3. **Economía**. Por ejemplo, imaginemos un sistema de información catastral en el que el usuario puede consultar los datos de una parcela concreta; utilizando un lenguaje interpretado del lado del cliente tendríamos que enviar toda la información del catastro para que éste luego extrajera la que le interesa. Obviamente esto también es un despropósito, pues tendríamos que enviar una cantidad ingente de datos a un cliente que está interesado sólo en una ínfima parte de ellos.

Y esto nos conduce al segundo actor de esta introducción: MySQL. Cuando un programador requiere almacenar una gran cantidad de datos en un archivo puede hacerlo utilizando alguno de los sistemas tradicionales (secuencial, aleatorio, …) y encargarse personalmente de programar los métodos necesarios para acceder, filtrar y modificar estos datos. Pero ¿por qué tomarse esta enorme molestia cuando existe MySQL, que un sistema de almacenamiento de datos muy eficiente y que además es compatible con el lenguaje de consultas estandarizado SQL?

MySQL es un sistema de gestión de bases de datos relacionales, que atesora los méritos de ser multiusuario y gratuito, pero cuya principal virtud es la compatibilidad con el lenguaje SQL (*Structured Query Language*).

Nota: Recordemos que las bases de datos están compuestas por tablas, que a su vez están compuestas por registros, que a su vez están compuestos por campos.

El lenguaje SQL sirve para consultar o modificar los datos de una base de datos utilizando un lenguaje de muy alto nivel (muy próximo a cómo nos expresamos los humanos). Por ejemplo, si quisiéramos conocer la calificación en la materia de matemáticas de un alumno llamado Javier podríamos utilizar una expresión similar a la siguiente:

```
001  SELECT campo_matematicas FROM tabla_calificaciones WHERE campo_alumno='Javier';
```

La sencillez del lenguaje SQL no le resta flexibilidad, pudiendo construirse consultas extremadamente complejas en las que, por ejemplo, se combinen campos de varias tablas filtrándolos a través de varios criterios y presentándolos ordenados de acuerdo a varios patrones.

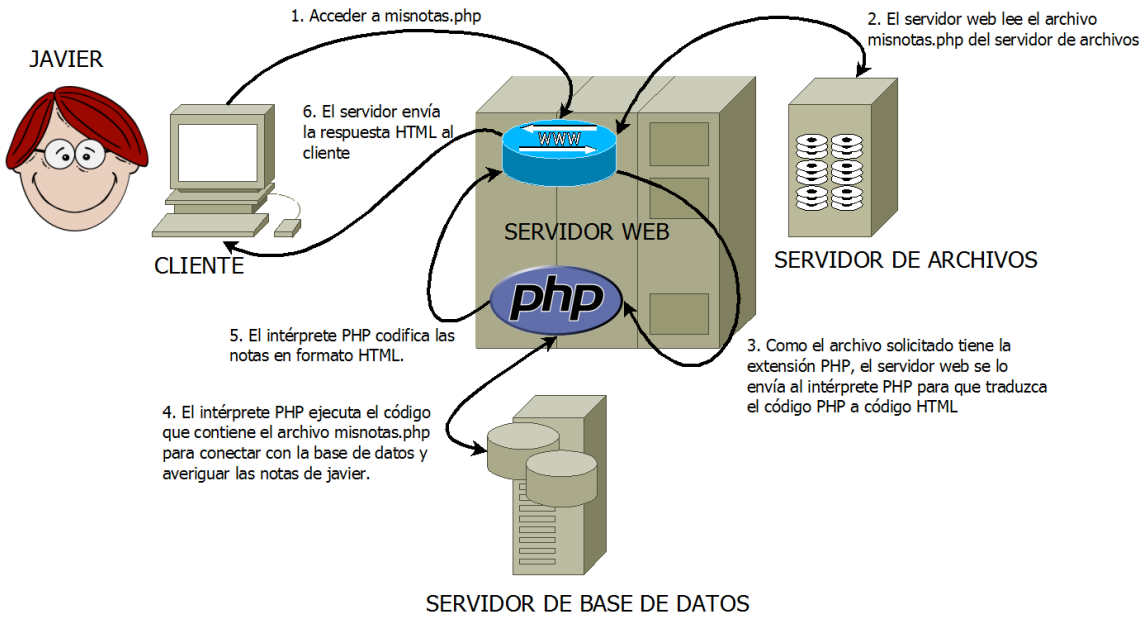
```
001  SELECT tabla_alumnos.campo_mombre, tabla_calificaciones.campo_materia,
     tabla_calificaciones.campo_calificacion
002  FROM tabla_alumnos LEFT JOIN tabla_calificaciones
003  ON tabla_alumnos.campo_nif=tabla_calificaciones.campo_nif
004  WHERE tabla_alumnos.curso='ESO PRIMERO'
005  GROUP BY tabla_alumnos.campo_nombre
006  ORDER BY tabla_alumnos.campo_nombre ASC;
```

Una vez elegidos los actores (PHP y MySQL) ya sólo nos queda rodar la película; y lo mejor de este trabajo es que va a resultar muy sencillo, pues PHP ha sido diseñado para facilitar la integración con MySQL, ofreciendo funciones muy cómodas para conectar con bases de datos MySQL y realizar consultas sobre ellas. La combinación entre PHP y MySQL conforma un tándem tan exitoso que ha

sido elegido por sistemas tan relevantes como:

- Wikipedia
- Drupal
- Joomla!
- Facebook
- WordPress
- Moodle

En la siguiente imagen se esquematiza el funcionamiento de PHP y MySQL.



Funcionamiento de un programa codificado en PHP.

Veamos los pasos que se dan desde que se inicia un programa en PHP hasta que se ven los resultados de su ejecución en la pantalla del cliente:

1. Se escribe el código fuente de la página web en lenguaje PHP.
2. Se guarda este código en el servidor web.
3. El navegador de un usuario solicita al servidor la página codificada y guardada en los pasos anteriores.
4. El servidor interpreta el código PHP.
5. El servidor envía el resultado del código PHP al navegador del usurario, que ve en su pantalla la página en formato HTML.

2. Breve historia de PHP

Antes de PHP, la única opción para insertar datos dinámicos en las páginas web desde el lado del servidor era recurrir a la tecnología CGI (*Common Gate Interface*), que consiste en que el servidor web encarga a una aplicación ejecutable (llamada CGI) la realización de ciertas tareas (por ejemplo, extraer datos de una base de datos). El problema de este método es que los CGI son aplicaciones ejecutables (generalmente desarrolladas en C), de modo que no son multi-plataforma y deben ser instaladas en el sistema. Esto suponía una importante limitación para los usuarios que no disponían de un servidor propio, pues era la empresa de hospedaje (*hosting*) la que decidía qué CGIs instalar y cuáles no (obviamente, por motivos de seguridad no se podía permitir que cualquier usuario instalase un archivo ejecutable en un servidor que era compartido por otros usuarios). Generalmente en los hospedajes compartidos sólo estaban disponibles CGIs muy básicos, como contadores de visitas, o muy genéricos, como recolectores de datos de formularios que eran enviados a una dirección de correo electrónico.

Ante esta situación, al ingeniero danés-canadiense **Rasmus Lerdorf** se le ocurrió desarrollar un CGI que funcionase como un intérprete genérico de instrucciones incrustadas en las páginas HTML (*scripts*). Esto suponía una revolución importante respecto a los CGIs tradicionales: el usuario podría incrustar el código en la página web para que fuese interpretado por el CGI de Lerdorf, de modo que ya no era necesario compilar un CGI específico para cada tarea ni instalarlo en el servidor, y además se lograba que ese código fuese independiente de la plataforma. La idea de Lerdorf alcanzó un estado de madurez suficiente en 1995, y la denominó PHP Tools (*Personal Home Page Tools*).

Lerdorf decidió distribuir su PHP Tools en formato de código abierto, de modo que otros programadores pudieran modificarlo o mejorarlo. Gracias a esta decisión los israelíes Andi Gutmans y Zeev Suraski pudieron reescribir algunas de las partes de PHP Tools (básicamente el intérprete sintáctico) para dotarlo de mayor robustez, dando lugar en 1998 a una versión nueva que denominaron PHP 3. En esta versión PHP es un acrónimo recursivo (tan populares en el mundo del software libre) que significa *PHP Hypertext Preprocessor*.

La mayor innovación de PHP 3 fue la posibilidad de ampliar sus posibilidades a través de módulos externos que cualquiera podría programar.

Zeev y Andi apostaron fuerte por este proyecto, procurando dotarlo de una mayor idoneidad para el sector profesional, y fundaron la empresa Zend, que reescribió todo el núcleo de PHP y lo denominó motor Zend (*Zend engine*).

Con este nuevo motor se lanzó en 2000 PHP 4, que mejoraba notablemente la eficiencia en el consumo de recursos y la velocidad de ejecución, y que además lograba una completa abstracción respecto al servidor web, de modo que PHP pasaba de poder funcionar sólo sobre el servidor web Apache a poder hacerlo prácticamente sobre cualquier servidor (Microsoft IIS y otros).

En 2004 se finalizó el desarrollo de la versión 2 del motor Zend, con el que se dotó a PHP 5, que es la versión vigente actualmente. PHP 5 introdujo la posibilidad real de utilizar programación orientada a objetos y mejoró la integración con el gestor de bases de datos MySQL.

Nota: En 2005 se anunció el inicio del desarrollo de PHP 6, cuya mayor novedad sería la adopción interna de la codificación Unicode. La implementación de esta mejora se ha revelado mucho más compleja de lo pensado

inicialmente, provocando que el lanzamiento de PHP 6 se haya retrasado en varias ocasiones y actualmente tenga una fecha de finalización indefinida. Los sucesivos anuncios de inmediato lanzamiento animaron a algunas editoriales a publicar libros con el título PHP 6. Para resolver este conflicto y sosegar a la comunidad de desarrolladores de PHP que estaba hastiada de tantos retrasos, en 2010 se decidió aislar la implementación de la codificación Unicode a una rama de desarrollo independiente, de modo que el resto de funcionalidades previstas para PHP 6 quedaron integradas en PHP 5.3. Consecuentemente, a excepción de lo referente a Unicode, los libros de PHP 6 corresponden a PHP 5.3. La última versión disponible en el momento de escribir estas líneas es la 5.3.6.

3. Breve historia de MySQL

En 1995 Michael Widenius, David Axmark y Allan Larsson lanzaron la primera versión de su gestor de bases de datos relacionales MySQL, que era inicialmente un producto comercial de código cerrado y fundaron la empresa MySQL AB.

Posteriormente, en 2000, decidieron convertirlo en un producto de código abierto con licencia GPL y, aunque inicialmente esto supuso una importante pérdida de ingresos para MySQL AB, ha terminado convirtiendo a MySQL en el gestor de bases de datos más popular del momento, compitiendo cara a cara con otros gestores comerciales.

En 2008 MySQL AB, fue adquirida por Sun Microsystems, que a su vez fue adquirida en 2010 por Oracle (el líder de facto en el mercado de las bases de datos relacionales). Fuertes discrepancias con el nuevo propietario, indujeron a los fundadores iniciales a abandonar el proyecto.

La versión más reciente de MySQL en el momento de escribir estas líneas es la 5.5.13.

4. Servidores web de desarrollo y explotación

Para crear un sitio web suelen utilizarse dos servidores: el de desarrollo y el de explotación.

El servidor de desarrollo es el que utilizan los programadores para desarrollar y depurar el código y, una vez que están convencidos de que funciona correctamente, lo trasladan al servidor de explotación.

Obviamente, es importante que el servidor de desarrollo tenga unas características idénticas o lo más similares posible al de explotación para evitar problemas de compatibilidad. Por ejemplo, si utilizamos como servidor de desarrollo uno dotado de PHP 5, pero el servidor de explotación utiliza PHP 4, es posible que nos encontremos con la desagradable sorpresa de que nuestro código no funcione, pues con cada nueva versión de PHP se desechan algunas características de la versión anterior por considerarse obsoletas o débiles desde el punto de vista de la seguridad.

Actualmente el servidor web más utilizado es Apache, sobre el que PHP puede instalarse de dos formas diferentes:

- Como un CGI. Independiza el servidor web del intérprete PHP, de modo que puede controlarse mejor la seguridad, pues el servidor web y el intérprete PHP tienen propietarios diferentes.

- Como un módulo del propio Apache. Es el método recomendado porque mejora la eficiencia (velocidad y consumo de recursos) y también el más popular.

Nota: Actualmente los servidores de EducaMadrid ejecutan PHP como un CGI, aunque la mayoría de los proveedores de hosting privados lo ejecutan como un módulo.

Como desarrolladores, en principio no debería afectarnos demasiado que PHP esté instalado como un módulo o un CGI, pero lo cierto es que dependiendo de la opción elegida por la empresa de hosting podemos encontrarnos con que ciertas funcionalidades de PHP estén deshabilitadas (generalmente por motivos de seguridad). Posteriormente en este mismo tema se explica cómo averiguar si PHP está instalado como un módulo o un CGI.

WAMP Server

Para configurar un ordenador normal como servidor web dinámico con acceso a bases de datos tendríamos que instalar e integrar en él los siguientes elementos:

- Una aplicación de tipo servidor web (la más popular es Apache).
- El intérprete PHP.
- El gestor de bases de datos relacionales (el más popular es MySQL).

Esta tarea puede resultar problemática en algunos ordenadores, principalmente en lo referente a la integración, pues hay multitud de detalles a tener en cuenta que varían dependiendo de las versiones de Apache, PHP y MySQL elegidas, y del sistema operativo del ordenador. Estos detalles generalmente se establecen mediante la edición manual de archivos de configuración, resultando bastante tedioso.

Para evitar estos problemas existen distribuciones que "empaquetan" estos 3 elementos (Apache, PHP y MySQL), además de otros, en un único archivo de instalación, de modo que su instalación resulta tan sencilla como la de cualquier aplicación de escritorio. Estas distribuciones se conocen con el nombre genérico de distribuciones LAMP, por las siglas de Linux, Apache, MySQL y PHP/Perl/Python/Postgres y algunas de las variantes más populares son:

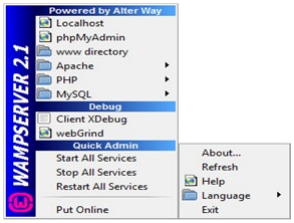
- XAMPP. Versión para Linux, Windows y MacOS.
- WAMP. Versión para Windows.
- MAMP. Versión para MacOs.

Nota: Muchas distribuciones de Linux incluyen los programas Apache, PHP y MySQL, de modo que no tenemos que preocuparnos de instalarlos personalmente. Por ejemplo, la distribución MAX Linux de la Comunidad de Madrid los incluye, y la distribución Ubuntu Server también.

Para este curso hemos elegido crear nuestro servidor de desarrollo sobre Windows, por considerarlo el sistema operativo más extendido, e instalaremos WAMP, que es posiblemente la solución de tipo LAMP más sencilla de instalar y flexible, pero antes se recomienda instalar el navegador Firefox, pues en uno de los pasos de instalación de WAMP nos preguntará qué navegador queremos usar de forma predeterminada y actualmente Firefox es el que más opciones nos ofrece a los desarrolladores.

Siga estos pasos para instalar WAMP en su ordenador:

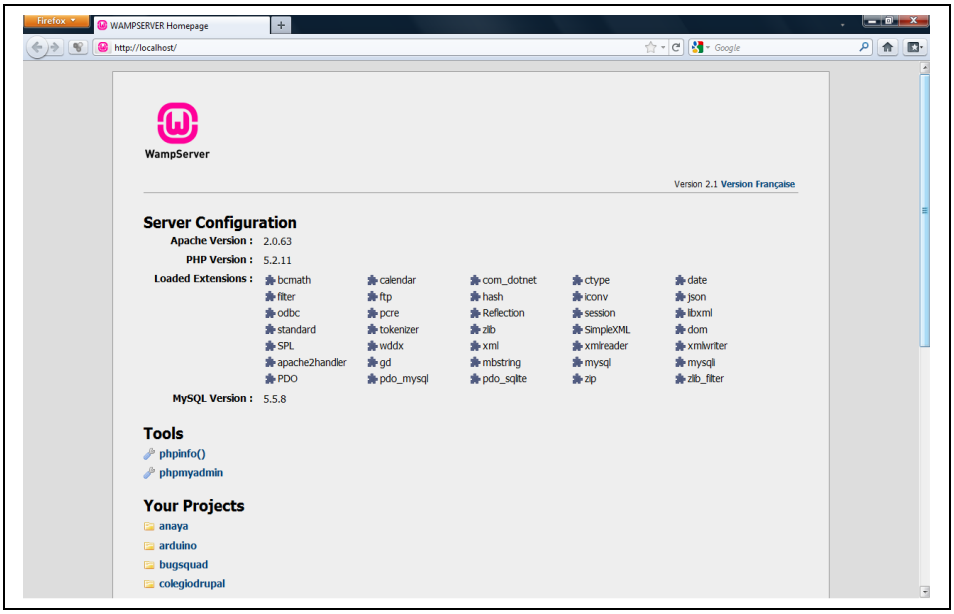
- Acceda a la dirección http://www.wampserver.com/ si prefiere la versión en francés, o a http://www.wampserver.com/en si entiende mejor inglés (en los siguientes paso se asume que ha optado por la versión en inglés).
- Haga clic sobre **Downloads** en la zona superior para acceder a la página de descargas.
- Existen versiones distintas de WAMP para equipos de 32 bits y 64 bits. Haga clic sobre la correspondiente a su ordenador. Si tiene dudas, elija la versión de 32 bits. Se descargará el archivo ejecutable de instalación.
- Ejecute el programa de instalación y siga los pasos del asistente (que básicamente se reducen a pulsar el botón **Next** aceptando todas las opciones predeterminadas, excepto la del navegador, en la que elegiremos el path de Firefox, que generalmente es C:\Program Files\Mozilla Firefox\firefox.exe.
- En el último paso del asistente mantenga activada la casilla que inicia WAMP y pulse el botón **Finish**. Fíjese en la bandeja de iconos (junto al reloj de la barra de tareas de Windows); verá aparecer una W, que es el icono que simboliza que WAMP está en funcionamiento. El color de este icono irá variando de rojo a naranja, y finalmente a verde, según se vayan ejecutando los servicios necesarios. Si no alcanza el color verde, el servidor no estará completamente operativo; esta situación prácticamente sólo se presenta por injerencias del Firewall o del antivirus o de algún otro servidor utilizando el puerto 80 (como Skype). Si se le presenta, consulte la documentación de su Firewall/Antivirus para obtener información sobre cómo permitir el funcionamiento de WAMP, o ejecute la opción **Apache>Service>Probar puerto 80** en el menú de WAMP para averiguar qué otras aplicaciones están usando el puerto 80.
- El icono W de WAMP nos presenta dos menús diferentes dependiendo de si hacemos clic sobre él con el botón izquierdo (véase la figura de la izquierda a continuación) o derecho del ratón (véase la figura de la derecha a continuación); generalmente utilizaremos el menú al que conduce el botón izquierdo (en este texto nos referiremos a él simplemente como menú de WAMP), aunque la opción de cerrar la aplicación (**Exit**) sólo está disponible en el del botón derecho; otra posibilidad interesante del menú al que conduce el botón derecho es elegir el idioma **spanish**.



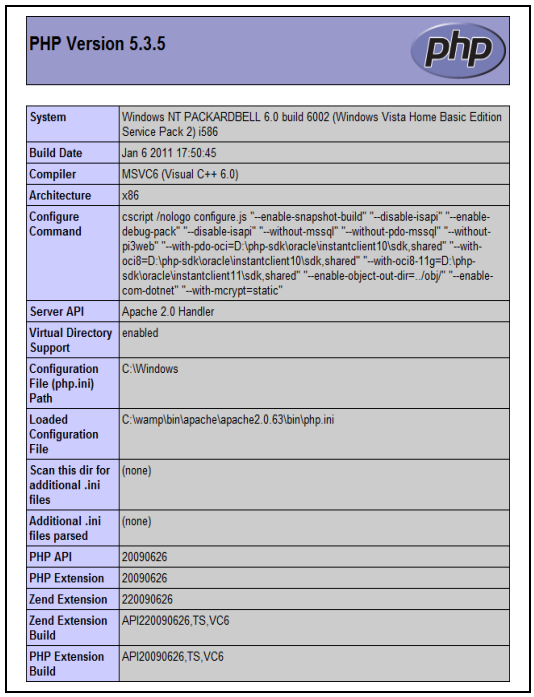
WAMP está configurado para utilizar la carpeta **c:\wamp\www** como ubicación predefinida de nuestros sitios web. Esta carpeta contiene algunos archivos propios de WAMP que facilitan el acceso a ciertas funciones esenciales y que es recomendable no borrar. Cada subcarpeta que creemos dentro de c:\wamp\www será considerada un proyecto diferente.

Por ejemplo, si seleccionamos **Localhost** en el menú del botón izquierdo de WAMP se iniciará nuestro navegador predeterminado (Firefox) mostrando la página de información general de WAMP (la visualización de esta página se debe a los archivos que se recomendaba no borrar en el párrafo anterior). En esta página podemos:

- Averiguar qué versión de Apache, PHP y MySQL estamos utilizando, y cuáles son los módulos (extensiones de PHP habilitados).
- Acceder a la función phpinfo(), que ofrece información más detallada sobre la configuración del intérprete PHP.
- Acceder a la aplicación web phpmyadmin, mediante la que es muy sencillo gestionar nuestras bases de datos.
- Acceder directamente a nuestros proyectos con un simple clic.



La opción **phpinfo()** ejecuta un instrucción de php llamada `phpinfo()` que muestra información detallada sobre nuestro servidor y el intérprete PHP (véase la figura siguiente).



PHP Version 5.3.5

System	Windows NT PACKARDBELL 6.0 build 6002 (Windows Vista Home Basic Edition Service Pack 2) i586
Build Date	Jan 6 2011 17:50:45
Compiler	MSVC6 (Visual C++ 6.0)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--disable-isapi" "--enable-debug-pack" "--disable-isapi" "--without-mssql" "--without-pdo-mssql" "--without-pi3web" "--with-pdo-oci=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8=D:\php-sdk\oracle\instantclient10\sdk,shared" "--with-oci8-11g=D:\php-sdk\oracle\instantclient11\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet" "--with-mcrypt=static"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\wamp\bin\apache\apache2.0.63\bin\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20090626
PHP Extension	20090626
Zend Extension	220090626
Zend Extension Build	API220090626,TS,VC6
PHP Extension Build	API20090626,TS,VC6

Una de las primeras informaciones que se muestra en phpinfo es **Server API**, mediante la que podemos saber si el intérprete PHP se está ejecutando como un CGI (su valor será CGI/Fast CGI) o como un módulo (su valor será Apache Handler).

Inicialmente, por motivos de seguridad, WAMP se ejecuta en modo Apagado, lo que quiere decir que sólo actuará como servidor para las peticiones que reciba desde el propio ordenador; en otras palabras: no atenderá peticiones procedentes de otros ordenadores que pudieran llegarle a través de la red. En inglés este estado se denomina *Offline*.

Mediante la última opción del menú de WAMP podríamos pasar a modo Encendido (*Online*), de modo que nuestro servidor sería *visible* desde la red y cualquiera podría acceder a nuestros proyectos.

5. La aplicación ¡Hola mundo! con PHP

Tradicionalmente, al comenzar el estudio de cualquier lenguaje de programación se utiliza el ejemplo "¡Hola Mundo!", que es precisamente el que desarrollaremos en los siguientes pasos dando así por finalizadas las cuestiones relacionadas con la configuración del servidor e inaugurando "oficialmente" nuestra singladura por el fascinante mundo de PHP:

1. Cree en c:\wamp\www la carpeta ut1 y dentro de ella un archivo llamado holamundo.php con el siguiente contenido.

```
001   <?php
002           echo "¡Hola
      mundo!";
003   ?>
```

2. Acceda a él con su navegador.
3. Compruebe las diferencias al cambiar la codificación del archivo holamundo.php entre ANSI y UTF-8 sin BOM (menú **Codificación>Convertir a** de Notepad++).
4. Compruebe que el agente de usuario (en nuestro caso Firefox) permite elegir e incluso intentar auto-detectar la codificación de la página (menú **Ver>Codificación de caracteres**).
5. ¿Cómo conjugar entonces la codificación real del archivo con la utilizada en el agente de usuario para que siempre se muestre correctamente? Existen dos opciones: indicar la codificación en la cabecera del propio archivo, o utilizar entidades HTML. Para poner en práctica la primera opción, añada a su página el código que se resalta en negrita a continuación si la tiene codificada en UTF-8. Si la tiene codificada en ANSI, sustituya charset=utf-8 por charset=iso-8859-1.

```
001  <html>
002          <head>
003                  <meta http-equiv="content-type" content="text/html;
     charset=utf-8">
004          </head>
005          <body>
007                  <?php
006                          echo "¡Hola mundo!";
007                  ?>
008          </body>
009  </html>
```

6. Cuando no nos es posible utilizar encabezados http para indicar la codificación (por ejemplo, si nuestro script genera un texto que puede incrustarse indistintamente en páginas con formato ANSI o UTF-8), la alternativa es utilizar entidades HTML. Una entidad HTML es una notación para referirse a un carácter; existen entidades HTML para todos los caracteres Unicode; las entidades HTML pueden utilizar referencia por nombre, como `&deg;` o por número, como `&#176;` (obsérvese la diferencia en la notación al añadir el signo #). Afortunadamente, existe un plugin para Notepad++ llamado HTML tag unicode con el que resulta muy sencillo convertir un texto seleccionado a entidades HTML. Descargue el plugin HTML tag de la dirección http://sourceforge.net/projects/npp-plugins/files/HTMLTag/ e instálelo copiando el archivo dll y el archivo ini en la carpeta plugins de Notepad++. Reinicie Notepad++. Seleccione el texto ¡Hola mundo! y ejecute el comando **Plugins>HTML tag>Encode HTML entities** (observe que el carácter ¡ se ha sustituido por ¡). Reduzca su página a sólo estas 3 líneas y pruebe que se muestra correctamente independientemente de cuál sea la codificación establecida en el agente de usuario (Firefox).

```
001  <?php
002          echo "&iexcl;Hola
     mundo!";
003  ?>
```

6. Necesito ayuda: php.net

Sin lugar a dudas, el mejor lugar para encontrar ayuda sobre PHP es el sitio php.net.

Este sitio contiene una referencia completa del lenguaje PHP, con explicaciones detalladas y ejemplos, pero posiblemente lo más interesante sean los comentarios que los propios programadores van añadiendo a la explicación de cada instrucción; al final de cada página hay una sección con las contribuciones realizadas por los programadores en las que es fácil encontrar soluciones a problemas más específicos (cuando se encuentre un problema, piense que posiblemente no sea el primero que se enfrenta a él…).

Observe también que en la zona superior izquierda hay un cuadro de lista que permite cambiar el idioma de la explicación, eligiendo por ejemplo Spanish.