

Hernán Beati

PHP

Creación de páginas Web dinámicas

2ª edición

Apoyo en la



PHP

Creación de páginas Web dinámicas

2ª ed.

Hernán Beati

PHP

Creación de páginas Web dinámicas

2ª ed.

Hernán Beati



Buenos Aires • Bogotá • México DF • Santiago de Chile

Beati, Hernán
PHP : creación de páginas web dinámicas . - 2a ed. - Ciudad Autónoma de Buenos Aires : Alfaomega Grupo Editor Argentino, 2015.
428 p. ; 23x17 cm.
ISBN 978-987-3832-04-8
1. Programación. I. Título
CDD 004.6

Queda prohibida la reproducción total o parcial de esta obra, su tratamiento informático y/o la transmisión por cualquier otra forma o medio sin autorización escrita de Alfaomega Grupo Editor Argentino S. A.

Edición: Damián Fernández
Corrección: Vanesa García
Armado de interiores: Vanesa García
Diseño de tapa: Diego Linares
Revisión de armado: Vanesa García

Internet: <http://www.alfaomega.com.mx>

Todos los derechos reservados © 2015, por Alfaomega Grupo Editor Argentino S. A.
Paraguay 1307, PB, oficina 11
ISBN 978-987-3832-04-8
Queda hecho el depósito que prevé la ley 11.723

NOTA IMPORTANTE: La información contenida en esta obra tiene un fin exclusivamente didáctico y, por lo tanto, no está previsto su aprovechamiento a nivel profesional o industrial. Las indicaciones técnicas y programas incluidos han sido elaborados con gran cuidado por el autor y reproducidos bajo estrictas normas de control. Alfaomega Grupo Editor Argentino S. A. no será jurídicamente responsable por: errores u omisiones; daños y perjuicios que se pudieran atribuir al uso de la información comprendida en este libro, ni por la utilización indebida que pudiera dársele. Los nombres comerciales que aparecen en este libro son marcas registradas de sus propietarios y se mencionan únicamente con fines didácticos, por lo que Alfaomega Grupo Editor Argentino S. A. no asume ninguna responsabilidad por el uso que se de a esta información, ya que no infringe ningún derecho de registro de marca. Los datos de los ejemplos y pantallas son ficticios, a no ser que se especifique lo contrario.
Los hipervínculos a los que se hace referencia no necesariamente son administrados por la editorial, por lo que no somos responsables de sus contenidos o de su disponibilidad en línea.

Empresas del grupo:

Argentina: Alfaomega Grupo Editor Argentino, S. A.
Paraguay 1307 P.B. "11", Buenos Aires, Argentina, C.P. 1057
Tel.: (54-11) 4811-7183/0887
E-mail: ventas@alfaomegaeditor.com.ar

México: Alfaomega Grupo Editor, S. A. de C.V.
Pitágoras 1139, Col. Del Valle, México, D.F., México, C.P. 03100
Tel.: (52-55) 5575-5022 – Fax: (52-55) 5575-2420/2490. Sin costo: 01-800-020-4396
E-mail: atencionalcliente@alfaomega.com.mx

Colombia: Alfaomega Colombiana S. A.
Calle 62 N° 20-46, Bogotá, Colombia
Tel. (57-1) 7460102 - Fax: (57-1) 2100415
E-mail: cliente@alfaomega.com.co

Chile: Alfaomega Grupo Editor, S. A.
Av. Providencia 1443, Oficina 24, Santiago de Chile, Chile
Tel.: (56-2) 235-4248/2947-5786 – Fax: (56-2) 235-5786
E-mail: agechile@alfaomega.cl

Mensaje del editor

Los conocimientos son esenciales en el desempeño profesional, sin ellos es imposible lograr las habilidades para competir laboralmente. La universidad o las instituciones de formación para el trabajo ofrecen la oportunidad de adquirir conocimientos que serán aprovechados más adelante en beneficio propio y de la sociedad; el avance de la ciencia y de la técnica hace necesario actualizar continuamente esos conocimientos. Cuando se toma la decisión de embarcarse en una vida profesional, se adquiere un compromiso de por vida: mantenerse al día en los conocimientos del área u oficio que se ha decidido desempeñar.

Alfaomega tiene por misión ofrecerles a estudiantes y profesionales conocimientos actualizados dentro de lineamientos pedagógicos que faciliten su utilización y permitan desarrollar las competencias requeridas por una profesión determinada. Alfaomega espera ser su compañera profesional en este viaje de por vida por el mundo del conocimiento.

Alfaomega hace uso de los medios impresos tradicionales en combinación con las tecnologías de la información y las comunicaciones (TIC) para facilitar el aprendizaje.

Libros como éste tienen su complemento en una página Web, en donde el alumno y su profesor encontrarán materiales adicionales.

Esta obra contiene numerosos gráficos, cuadros y otros recursos para despertar el interés del estudiante, y facilitarle la comprensión y apropiación del conocimiento. Cada capítulo se desarrolla con argumentos presentados en forma sencilla y estructurada claramente hacia los objetivos y metas propuestas.

Los libros de Alfaomega están diseñados para ser utilizados dentro de los procesos de enseñanza-aprendizaje, y pueden ser usados como textos para diversos cursos o como apoyo para reforzar el desarrollo profesional.

Alfaomega espera contribuir así a la formación y el desarrollo de profesionales exitosos para beneficio de la sociedad.

Acerca del autor

No nací programador. Por esta razón, estoy convencido de que todas las personas pueden aprender a programar y me especializo en su capacitación.

Al finalizar mi adolescencia, comencé mi formación pedagógica cursando un magisterio de música (que nunca ejercí); pero paralelamente me dediqué profesionalmente al diseño gráfico durante casi una década, y ya hacia fines del siglo XX, me convertí en diseñador Web.

Conocer el lenguaje PHP en el año 2000 fue lo que me hizo estudiar Análisis de Sistemas y la Licenciatura en Informática Educativa y pronto, me especialicé en programación para la Web, realizando numerosas aplicaciones Web, trabajando tanto para empresas como freelance.

Desarrollé mi perfil docente desde el año 2002, cuando fundé el campus virtual de SaberWeb y comencé a enseñar en los institutos ITMaster, Image Campus y Dotzero, en las carreras terciarias de Escuela Da Vinci e IMAGE, y para graduados de las carreras de Diseño en la FADU de la Universidad de Buenos Aires. En la Universidad Tecnológica Nacional (FRBA), dicté el curso Professional Webmaster y fui el creador del curso de Programador Web Avanzado, que aún se sigue dictando.

Desde 2013, soy profesor titular de las materias Programación Multimedial II y III en la Universidad Maimónides, dentro de la Licenciatura en Tecnología Multimedial, donde investigamos sobre la base de PHP y MySQL la creación de servicios Web para Apps móviles y videojuegos, y creamos aplicaciones Web orientadas a objetos.

Sigo capacitándome continuamente: los últimos cursos que hice en esta área fueron sobre PHP orientado a objetos, y backend de aplicaciones móviles y videojuegos.

Siempre me gustó el modelo de desarrollo colaborativo, propio del software libre, del que PHP es un perfecto exponente.

Hernán Beati, julio de 2015

*A mis cuatro mujeres (mi esposa Katty y mis hijas Stephanie, Violeta y Mora),
que soportaron pacientemente las horas que les robé
para poder escribir este libro.*

Hernán Beati

Agradecimientos

A Maximiliano Firtman, que siempre confió en mis posibilidades, haciéndome parte de sus equipos de docentes, y ahora brindándome la oportunidad de publicar este libro.

A Damián Fernández, de Alfaomega Grupo Editor, por su seguimiento y estímulo constante.

A todos mis alumnos, porque todo este esfuerzo solo cobra sentido en la medida en que puedan aprovecharlo para su crecimiento profesional.

Hernán Beati

Contenido

Prólogo.....	XIII
Plataforma de contenidos interactivos.....	XVI

CAPÍTULO 1	
MÁS ALLÁ DE HTML Y CSS.....	1
¡No más páginas Web: aplicaciones Web.....	1

CAPÍTULO 2	
EL AMBIENTE PHP.....	15
Esas extrañas siglas: LAMP, MAMP,	
WAMP, xAMP.....	15

CAPÍTULO 3	
MEZCLANDO PHP Y HTML.....	35
El concepto clave: completando las	
páginas HTML en el acto	35

CAPÍTULO 4	
LOS ALMACENES DE DATOS	51
Contenedores temporales y	
permanentes, de pocos y de muchos	
datos.....	51
Las variables: pocos datos, provisorios	53
Las constantes: pocos datos que no	
cambiaremos.....	65
Las matrices: muchos datos provisorios....	69

CAPÍTULO 5	
ENVIANDO DATOS HACIA EL	
SERVIDOR.....	81
Herramientas para enviar datos: enlaces	
y formularios.....	81

CAPÍTULO 6	
VALIDACIONES.....	91
Validando datos de formularios y enlaces.	91
Las condicionales.....	92

¿Qué hacer si responde que no es	
verdad? El else y el elseif.....	94

CAPÍTULO 7	
IDENTIFICACIÓN CON COOKIES Y	
SESIONES	123
Cookies: datos que identifican a un	
navegador	123
Sesiones: datos que identifican a un	
usuario.....	141

CAPÍTULO 8	
LOS BUCLES Y LOS ARCHIVOS DE	
TEXTO	163
Recorriendo línea por línea la información	
información almacenada	163
Tipos de bucles: for, while, do while,	
foreach.....	164
Condicionales dentro de bucles	179
Los archivos de texto.....	182
Formas de leer datos desde un archivo	
de texto.....	186
Cómo escribir y acumular datos en un	
archivo de texto	198

CAPÍTULO 9	
CREANDO Y USANDO FUNCIONES	203
Reutilizando nuestros códigos.....	203
Planificando nuestros sistemas Web.....	205
La función: una caja cerrada que procesa	
datos.....	207
Declarar una función.....	210

CAPÍTULO 10	
FUNCIONES INCORPORADAS MÁS	
USADAS	229

Manejo de caracteres, fechas y envío de correos	229
Funciones de manejo de caracteres	230
Funciones de fecha y hora	249
Funciones de envío de correos electrónicos.....	257

CAPÍTULO 11

CREANDO BASES DE DATOS

El almacén de datos más potente para nuestros sitios Web.....	269
Diferencia entre archivos de texto y bases de datos: el lenguaje SQL.....	270
Conceptos fundamentales: base, tabla, registro y campo.....	272
Creando bases y tablas con phpMyAdmin.....	275
Proceso de altas, bajas y modificaciones ..	289
Los tipos de datos más usados.....	297
Atributos de los campos	309

CAPÍTULO 12

LLEVANDO DATOS DE LA BASE A

LAS PÁGINAS

Cómo leer datos desde una base con PHP.....	315
PHP	315
Complementos de la orden SELECT del lenguaje SQL.....	324
Funciones propias para mostrar datos	331

CAPÍTULO 13

LLEVANDO DATOS DE LAS PÁGINAS

A LA BASE

Cómo escribir datos en una base desde PHP	345
Cómo eliminar datos de una base con PHP	357
Cómo modificar datos de una base con PHP	365
Radiografía de un sistema con back-end y front-end.....	379

APÉNDICE

PROGRAMACIÓN ORIENTADA A

OBJETOS.....

APÉNDICE WEB

ADAPTANDO SOFTWARE LIBRE

<http://libroweb.alfaomega.com.mx>

Prólogo

En 1994, un programador nacido en Groenlandia, llamado Rasmus Lerdorf (<http://lerdorf.com>), desarrolló un código que le ayudaría a crear su página Web personal de manera más sencilla. Lo llamó Personal Home Page Tools (PHP Tools) o herramientas para páginas iniciales personales. De las primeras tres palabras en inglés surge el nombre del lenguaje que finalmente se liberó al público, gratis, en 1995.

Quince años después, el mundo de la Web ha cambiado drásticamente. La evolución y difusión de PHP en el mundo del desarrollo Web ha ido mucho más allá de lo que Rasmus pudo imaginar; se trata de un mundo del que ahora podrás ser parte.

Dos años más tarde, junto a otras personas, se reescribe parte del código del lenguaje y se lanza la versión de PHP que ha llevado el lenguaje al estrellato: PHP 3. Un lenguaje simple, rápido y dinámico que permite crear páginas Web interactivas con muy poco código.

En ese momento deciden que el nombre Personal Home Page ya le quedaba un poco corto al lenguaje y deciden cambiar el significado de las siglas. Así es que hoy PHP significa “PHP Hypertext Preprocessor”. No es un error de imprenta: la “pe” de PHP significa PHP. Es una sigla recursiva (un truco de programadores) y el resto del nombre significa “pre-procesador de hipertexto”. Es un pre-procesador porque se ejecuta antes que el navegador y trabaja principalmente sobre hipertexto, que es el concepto subyacente de los documentos HTML.

Con los años igualmente se ha ganado su propio nombre. PHP es, simplemente, PHP. Es tan importante en la Web como lo es HTML. Es un lenguaje fácil de aprender, simple de usar, potente, rápido, gratuito, de código abierto y utilizado en más de la mitad de todos los sitios Web del mundo.

Solo para ejemplificar la potencia del lenguaje, mencionaremos que el sitio más importante y con más visitas hoy en el mundo, Facebook, está desarrollado con PHP.

Tuve la oportunidad de conocer personalmente a Rasmus en Madrid en una conferencia sobre optimización extrema de PHP y me ha quedado muy presente

una anécdota que me gustaría compartir contigo. Rasmus tuvo la oportunidad de analizar el código fuente PHP de las primeras versiones de Facebook. El código era un desastre, pésimamente programado, con errores por todos lados.

Y a pesar del desastre que él detectó que era esa Web desde el lado del código, ¡funcionaba bien! Y miren en lo que Facebook se ha convertido: una empresa billonaria. La moraleja de la anécdota es que la gran ventaja de PHP es su capacidad de funcionar sin problemas en cualquier circunstancia y de poder tener una Web lista en muy poco tiempo.

Rasmus, un poco exagerando para hacer entender el objetivo de su moraleja, comentó que los proyectos más exitosos en la Web no son los mejor programados, los que siguen las mejores prácticas o los que son desarrollados por académicos. Son aquellos que implementan las mejores ideas y lo hacen lo más rápido posible; por eso PHP es el lenguaje ideal para la Web.

Luego de esta introducción, estarás seguro con muchas ganas de empezar a trabajar con PHP. Y qué mejor que hacerlo de la mano de Hernán Beati, un excelente profesional y profesor que conozco hace ya diez años y recomiendo plenamente para tu viaje en el mundo de este apasionante lenguaje.

¡A programar!

Lic. Maximiliano Firtman
Director ITMaster
@firt



Alfaomega e ITMaster Professional Training te dan la posibilidad de que certifiques tus conocimientos y experiencias adquiridos como lector de este libro. Su aprobación te permitirá tener la certificación en Programación PHP Inicial.

Luego de la obtención del certificado, podrás continuar tu formación en la carrera corta de Programador Web, en los másters de especialización Programador Experto PHP y Experto en Mobile Web y en los cursos cortos presenciales y online disponibles en todo el mundo.

Para dar la evaluación de certificación o recibir mayor información sobre este servicio, ingresa en el sitio Web o envíanos un e-mail a la dirección correspondiente a tu país.

España

<http://libros.itmaster.es> - info@itmaster.es

México

<http://libros.itmaster.com.mx> - info@itmaster.com.mx

Argentina y otros países

<http://libros.itmaster.com.ar> - info@itmaster.com.ar

Plataforma de contenidos interactivos

Para tener acceso al material de la plataforma de contenidos interactivos del libro: *PHP: creación de páginas Web dinámicas*, 2ª edición, siga los siguientes pasos:

1. Ir a la página: <http://libroweb.alfaomega.com.mx>
2. Ir a la sección *Catálogo* y seleccionar la imagen de la portada del libro, al dar doble clic sobre ella, tendrá acceso al material descargable.

NOTA: Se recomienda respaldar los archivos descargados de las páginas web en un soporte físico.

Más allá de HTML y CSS

1

¡No más páginas Web: aplicaciones Web!

“No es la Programación, son los Negocios”. Eso es PHP.

Dominar el lenguaje PHP amplía nuestros horizontes profesionales como diseñadores o programadores, y nos convierte en creadores de **Aplicaciones Web**. Nos lleva de la mano a un mundo de comercio electrónico, redes sociales, intranets, portales de noticias y entretenimientos, un mundo “mágico” en el que podemos acceder gratuitamente a miles de sistemas completos prearmados, listos para usar (y para vender a nuestros nuevos clientes).

Nos abre un nuevo mercado, donde los clientes ya no están tan interesados en el diseño (aunque puede aportar su encanto), sino en las funcionalidades, para que, a través de un navegador, las personas puedan hacer alguna tarea concreta en su sitio Web.

Ganando nuevos mercados a dos competidores: diseñadores gráficos y empresas de sistemas

PHP nos despega de nuestros antiguos competidores (diseñadores gráficos o programadores de aplicaciones de escritorio) y nos lleva a un nuevo mercado, en el que ofrecemos soluciones Web a comercios, empresas de diversos tamaños, profesionales, instituciones educativas, medios de difusión.

Ahora competimos con empresas de sistemas, con la ventaja de la rapidez y economía de nuestras soluciones PHP.

LENGUAJES	HTML/CSS	PHP/MySQL	Otros (Java, .Net)
COMPETIDORES	Diseñadores gráficos	Programadores Web	Empresas de sistemas
TAREA PRINCIPAL	Decorar páginas (no saben programar)	Adaptar sistemas pre-armados rápida y económicamente	Hacer sistemas a medida desde cero (caros y de largo plazo)
NIVEL DE PRESUPUESTOS	Cientos	Miles/Decenas de miles	Decenas a centenas de miles

Tabla 1-1. Mercado de los sistemas prearmados

Eso explica por qué se propaga con tanta velocidad el conocimiento de PHP entre diseñadores y programadores de otros lenguajes. Es la clave para llevar a cabo cualquier proyecto que trascienda las páginas Web HTML estáticas.

Y el detalle fundamental: con PHP, nuestros presupuestos pueden llegar a tener uno o incluso dos ceros más que los presupuestos que hacíamos como diseñadores o programadores de otros lenguajes.

Además, PHP es fácil de aprender.

La lógica de PHP: un amigo invisible

¿Qué es PHP? PHP no se ve. Es “transparente”, invisible. Por esta razón, es difícil explicar qué es y cuál su funcionamiento. Sin embargo, lo intentaremos.

Éste es el concepto más abstracto del libro, pero es imprescindible para entender qué hace PHP.

PHP es un acrónimo de *PHP: Hypertext Preprocessor*, es decir, “Preprocesador de Hipertexto marca PHP”. El hecho de que sea un preprocesador es lo que marca la diferencia entre el proceso que sufren las páginas Web programadas en PHP del de aquellas páginas Web comunes, escritas solo en lenguaje HTML.

Para llegar a entender qué es un preprocesador, examinaremos primero cuál es la diferencia entre el proceso de una página Web normal (HTML) y el preproceso de una página escrita en lenguaje PHP.

Proceso de archivos HTML

¿Cuál es el camino que sigue una página Web común (escrita en lenguaje HTML) desde que escribimos su dirección en nuestro navegador hasta que la vemos en nuestra pantalla?

- Comenzamos escribiendo en el navegador la URL deseada y pulsamos **enter** (o pulsamos un enlace con el *mouse*); en ambos casos, la barra de direcciones nos muestra la URL del archivo HTML que nuestro navegador está solicitando:

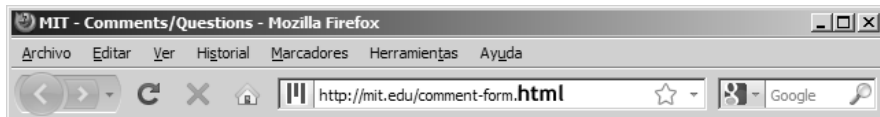


Figura 1-1. URL de un archivo HTML.

- En ese momento, el navegador envía una petición que solicita esa página. Ese pedido “viaja” desde nuestro navegador hasta la máquina *hosting* que hospeda el archivo requerido. Pero el navegador no remite únicamente el pedido del archivo que necesita, sino que lo acompaña con un número que nos identifica inequívocamente: nuestra dirección IP.



Figura 1-2. El navegador solicita un archivo y envía nuestra dirección IP.

- Podemos compararlo con un *delivery* de pizza a domicilio; para recibir el pedido, le decimos al telefonista dos datos: “cuál gusto de pizza” queremos y “a qué dirección” nos la debe enviar.
- Cuando el pedido llega al *hosting* indicado, un programa denominado servidor Web que está encendido en esa máquina, recibe el pedido y va a buscar el archivo solicitado en el disco rígido.



Figura 1-3. El servidor Web busca en el disco rígido del *hosting* el archivo solicitado.

- El rol del servidor Web es similar al del empleado de la pizzería que atiende al teléfono y va a buscar el pedido a la cocina de la pizzería.
- Ese servidor Web, una vez que localizó el archivo solicitado, envía, entrega o “sirve” (de ahí, su nombre: “servidor”) el archivo al navegador que se había quedado esperando una respuesta en la dirección IP que lo identifica.



Figura 1-4. El servidor Web envía el archivo solicitado a la dirección IP del navegador.

- Equivaldría al viaje de la moto que nos trae la pizza hacia nuestra casa.
- Una vez que llegó el archivo hasta nuestro navegador, éste se encarga de interpretar los contenidos de ese archivo de texto y código HTML, armando cada elemento (textos, tablas, colores) de la página recibida en nuestra pantalla para que la podamos leer.

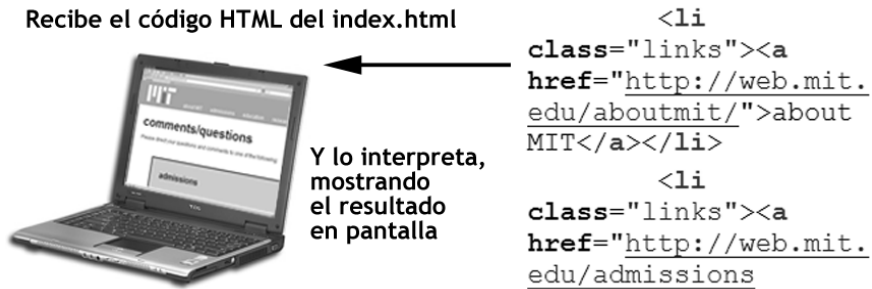


Figura 1-5. El navegador interpreta el código HTML y muestra el resultado.

En la metáfora de la pizza, llega el momento de recibir la pizza y servirla en nuestra mesa, lista para comer.

Podemos mencionar algunas conclusiones que identifican el proceso de archivos HTML comunes:

- Nuestro **navegador** tiene la capacidad de pedir archivos a distintos servidores Web, y a su vez, de entender y “descifrar” esos archivos recibidos, cuyo contenido es código HTML. Nuestro navegador es un programa que todo el tiempo realiza pedidos de archivos (peticiones) y recibe un archivo HTML como respuesta, que luego muestra a los seres humanos. Pide archivos y los muestra, pide y muestra...
- El **servidor Web** es un programa instalado en los *hostings*, que todo el tiempo recibe pedidos de navegadores (se los llama peticiones), y le entrega a esos navegadores el archivo HTML solicitado.
- Este diálogo entre un navegador y un servidor Web sigue las reglas de un **protocolo** (una convención, un estándar) denominado HTTP (*HyperText Transfer Protocol* o Protocolo de Transferencia de Hipertexto).

Todo esto sucede cada vez que queremos ver un archivo HTML común.

Preproceso de archivos PHP

Pero, ¿cuál es el camino que sigue una página Web cuya extensión es .php desde que escribimos su dirección en nuestro navegador hasta que la vemos?

Cuando la extensión del archivo solicitado es .php, se introduce un elemento diferente en este circuito:

- Hacemos el pedido de ver una página con extensión .php desde nuestro navegador:



Figura 1-6. URL de un archivo PHP.

- El programa servidor Web instalado en el *hosting* recibe nuestro pedido y, de inmediato, detecta que el archivo solicitado tiene extensión `.php` y, por lo tanto, deriva el pedido a otro programa que está encendido en esa misma máquina *hosting*, que se denomina **intérprete de PHP** (es una especie de “ser mágico”, cuya presencia es muy difícil intuir, y que debemos acostumbrarnos a imaginar que “está ahí” para poder programar correctamente en PHP).



Figura 1-7. El servidor Web le pasa el pedido al intérprete de PHP.

- Este programa intérprete de PHP busca en el disco rígido del *hosting* el archivo `.php` que fue solicitado, y comienza a leer su código línea por línea, buscando determinadas “marcas” o etiquetas que nosotros, como programadores, hemos dejado escritas y que contienen órdenes destinadas a ese programa intérprete de PHP.



Figura 1-8. El intérprete de PHP busca las marcas con órdenes para él.

- Cuando este programa intérprete de lenguaje PHP encuentra estas órdenes, las ejecuta (las procesa) y, a continuación, reemplaza todas las

órdenes que hubiese entre la apertura y el cierre de la etiqueta de PHP por el resultado de procesar esas órdenes. Es decir, **borra** las órdenes del código HTML en el que estaban escritas y, en su lugar, coloca los **datos** obtenidos como consecuencia de la ejecución de esas órdenes.

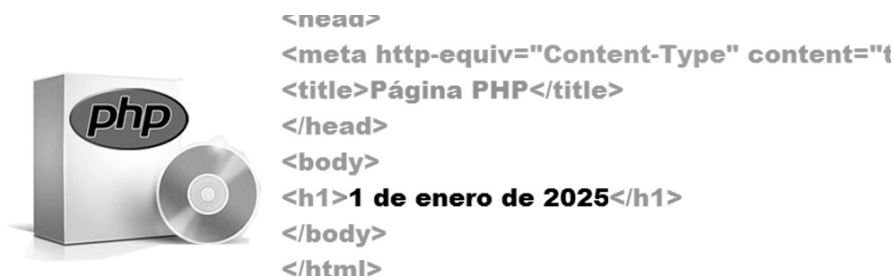


Figura 1-9. El intérprete de PHP borra cada orden y en su lugar escribe el resultado de procesar esa orden.

Veamos un ejemplo: supongamos que en una de esas órdenes le dijimos al intérprete de PHP que averigüe la fecha actual y que la escriba en el lugar exacto donde dejamos la orden escrita. Ésta se borrará del código fuente y, en su lugar, **quedará escrita la fecha**.

No se escribe así de simple y en español, pero ése es el concepto. Por esta razón, no debemos ejecutarlo porque no funcionará. Lo mencionamos, simplemente, para ilustrar la idea.

```

...
<body>
<h1>
<?php
Atención, software de PHP!:
Quiero que escribas la fecha actual aquí!
?>
</h1>
</body>
...

```

- El software de PHP ejecuta la orden que le dejamos escrita y, al finalizar, devuelve al software servidor Web **el texto y el código HTML producido**, para que el servidor Web lo entregue **al navegador**, que lo interpreta como si este código HTML, que incluye la fecha, hubiese estado escrito allí desde un principio:


```
...
<body>
<h1>1 de enero de 2025</h1>
</body>
...
```

- En el código fuente que le llega al **navegador**, no vemos ningún rastro de la orden que habíamos escrito para el software de PHP, ya que este software se ocupó de borrarla para que nadie la vea, y en el lugar exacto en el que habíamos escrito esa orden, colocó “el resultado de ejecutar esa orden”, es decir, la fecha, que se ocupó de conseguir.

En resumen, el **preproceso** de páginas PHP consiste en esta serie de pasos: dejamos escritas entre medio de nuestras páginas algunas **órdenes** destinadas al **software intérprete de PHP** (órdenes que casi siempre consisten en que el software de PHP obtenga cierta información, como la fecha del ejemplo anterior); luego, colocamos otras órdenes para que el software intérprete de PHP “realice algo” con esa información, típicamente, que la **escriba** dentro del código fuente de la página HTML que se enviará al navegador del usuario.

En la Figura 1-10, observamos algunas órdenes en lenguaje PHP escritas en medio del código HTML de una página:

```
</head>
<body>
<h1><?php echo $fecha; ?></h1>
<h2><?php echo $titulo; ?></h2>
<p><?php echo $noticia; ?></p>
</body>
</html>
```

Figura 1-10. Ejemplos de órdenes PHP intercaladas en el código HTML.

Páginas estáticas

De la diferencia entre los procesos que sufren las páginas HTML comunes y las páginas PHP, podemos concluir que las páginas Web, escritas únicamente en lenguaje HTML, son **estáticas**: es decir, **nunca cambian** su contenido: pase lo que

pase, lo que llegará al navegador del usuario es lo que ha sido escrito en ellas por el diseñador Web, ni más ni menos, siempre lo mismo.

Páginas dinámicas

Por el contrario, las páginas que incluyen código escrito en lenguaje PHP, nos dan la oportunidad de **personalizar** su contenido sobre la base de ciertas órdenes escritas (como en el ejemplo anterior de la fecha, la página hoy mostrará una fecha y mañana otra, y así sucesivamente; es decir: siempre generará un **contenido distinto**, variable).

El contenido de esas páginas, al menos en partes de ellas, cambiará y no será siempre el mismo, ya que dependerá de la información que obtenga el software de PHP y coloque en ellas. Serán páginas dinámicas.

Las bases de datos

El concepto de páginas dinámicas que acabamos de esbozar, se complementa a la perfección con las **bases de datos**, ya que éstas se ocupan de almacenar datos y, las páginas dinámicas, de **leerlos** y **mostrarlos** dentro de ellas.

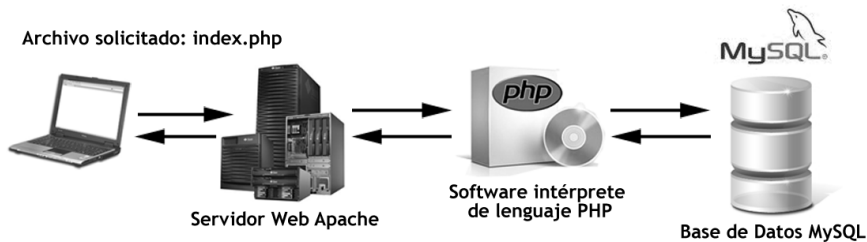


Figura 1-11. Esquema de interacción entre los programas que participan cuando las páginas utilizan una base de datos.

MySQL: la base de los proyectos Web exitosos

Desde la aparición de PHP, la base de datos que siempre estuvo asociada a PHP fue MySQL, no solo gracias a su gran potencia y rapidez, sino, fundamentalmente, a que su licencia no tenía costo para los *hostings* (a diferencia de Oracle, SQL Server y otros competidores), lo que contribuyó a su rápida difusión y a su mejoramiento vertiginoso en sucesivas versiones, que contribuyeron a que se convierta en la base de datos que provee almacenamiento a sistemas de la envergadura de Google, Facebook y millones de otros populares sitios Web en todo el mundo.

Pero, ¿para qué sirve usar una base de datos en un sitio Web? Veamos algunos de sus posibles usos.

Un camino de ida: del almacén a las páginas

Pensemos en un ejemplo: una página que muestra un catálogo de libros:



Figura 1-12. Página que muestra información de una base de datos.

Si pretendiéramos hacer este tipo de sitios únicamente con HTML, deberíamos crear esta página manualmente y, en caso de alguna modificación en los datos de alguno de los productos (un nuevo producto, o quitar uno existente), deberíamos modificar el archivo HTML a mano, y volver a colocar en el *hosting* una copia actualizada de esta página HTML mediante FTP. Así, en todos los cambios de cualquiera de los miles de artículos que se muestran.

En cambio, con PHP y MySQL, podemos crear una **base de datos** que contenga información sobre los miles de productos que se mostrarán. Luego, utilizaremos una **página dinámica** escrita en PHP, que incluya la orden de leer esa base de datos y publicar los productos. Si modificamos los datos, éstos se reemplazarán directamente en la base de datos, sin necesidad de modificar ni una línea de las páginas Web que exhiben los productos, ya que la orden sigue siendo exactamente la misma: “leer la base y mostrar todos los productos”.

Otra vez, para graficar la idea, la simplificaremos, pero pronto lo aprenderemos a programar para que funcione:

```
...  
<p>  
<?php  
Atención, software de PHP!:  
Quiero que le pidas a la base de datos la lista de  
productos y la muestres aquí!  
>  
</p>  
...
```

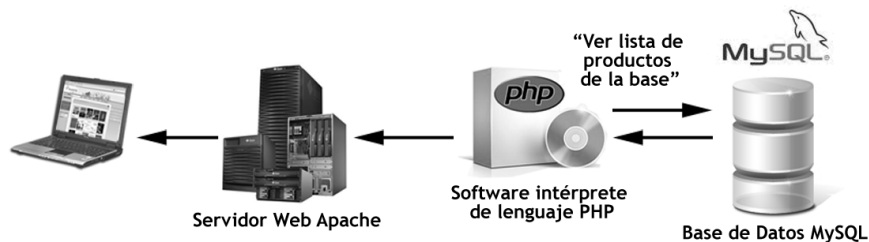


Figura 1-13. Esquema del circuito de una página que muestra información almacenada en una base de datos.

Un camino de vuelta: recepción de datos de formularios

Pero aquí no terminan las ventajas de almacenar en una base de datos la información de una página, sino que PHP y MySQL agregan un camino imposible de lograr solamente con HTML: la posibilidad de enviar datos desde el navegador del usuario hacia el servidor, y que estos datos queden almacenados en la base de datos del *hosting*.

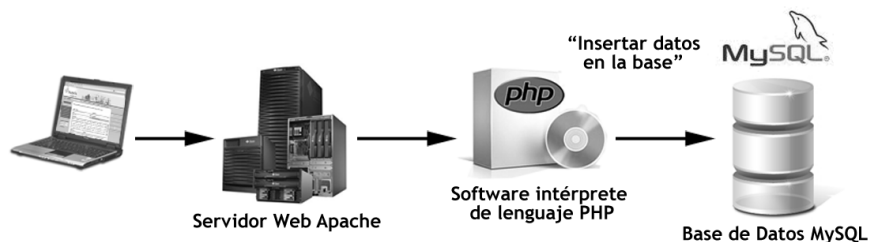


Figura 1-14. Esquema del circuito de una página que envía datos hacia el servidor.

¿Posibles usos?: envío de formularios con consultas, currículums, formularios para paneles de administración que actualizan datos de catálogos, envío de mensajes para publicar en foros, envío de fotografías, etc.

PAIDÓS
Grupo Planeta

¿COMO BUSCAR?

LANZAMIENTOS

Ud. puede modificar todos los datos de este Libro, salvo su código, y salvo su tema

Código: Título: Subtítulo:
10036 SIMBIOSIS Y AMBIGÜEDAD Estudio psicoanalítico

Autor: Tema: Colección:
Bleger, José 2 PSICOLOGÍA PROFUNDA

Descripción:
Este libro parte de la hipótesis de que el núcleo interdependencia simbiótica es de carácter amb relación de la simbiosis con el proceso de pro y las modalidades de la relación objetual, y se estructura de la ambigüedad en la clínica. A t la ambigüedad, se reconsideran conceptos como

Subgénero: Público:
Teoría y clínica Universitario y Profesional

ISBN: Medida: Páginas: Código de Barras:
950-12-4036-3 13,5 x 22 320 9 789501 240368

Mes: Año:
Diciembre 2003

[- Volver al Panel -](#)

Figura 1-15. Ejemplo de formulario que envía datos al servidor y los guarda en una base de datos.

Sitios autoadministrables: CMS con Front-end y Back-end

La combinación de los dos “camino” anteriores (enviar datos **hacia** la base de datos y, luego, **mostrar** esos datos mediante páginas dinámicas) permiten crear sistemas denominados CMS (*Content Management System* o Sistemas de Administración de Contenidos) que gestionan la publicación de contenidos de portales, sistemas de comercio electrónico, comunidades *on-line*, y muchas otras aplicaciones, siempre sobre la base de esos conceptos básicos: el usuario envía datos hacia el *hosting*, que se almacenan en la base de datos y, luego, esos datos se muestran (a él o a otros usuarios autorizados).

Se denomina *Front-end* a la parte visible por los usuarios, y *Back-end* al panel de administración de acceso restringido solo a los usuarios autorizados (como el de la última imagen).

Millares de CMS de uso libre y gratuito

Existen literalmente miles de CMS con distintas finalidades, listos para usar, con licencia de libre uso y modificación (típicamente, la licencia GPL y similares), creados con PHP y MySQL. Algunos de los más famosos son *Joomla*, *WordPress*, *osCommerce*, *Mambo*, *Drupal*, *Moodle*, *Magento*, *OpenRealty*, pero existen miles más.

El secreto del negocio está en **saber programar en PHP** lo suficiente como para **entender** y poder realizar algunas **modificaciones** para adaptar estos CMS a las necesidades específicas de cada cliente, y poder ofrecerles soluciones de bajo costo y muy rápida implementación.



Figura 1-16. Nuestro negocio: tomar código prearmado y modificarlo según las necesidades de nuestros clientes.

De aprender lo necesario de PHP y MySQL para poder hacer esas modificaciones, es de lo que nos ocuparemos en el resto de este libro.

El ambiente PHP

2

Esas extrañas siglas: LAMP, MAMP, WAMP, xAMP

Hojeando libros o mirando tutoriales en la Web, encontraremos muchas veces asociada a PHP alguna de estas siglas. ¿A qué se refieren?

Es muy simple: si comenzamos por el final, la “P” corresponde a **PHP**, la “M”, a **MySQL** y, la “A”, a **Apache**, que es el servidor Web más usado en los *hostings* que tienen instalado el intérprete de PHP.

¿Y esa primera letra, la “L”, “M”, “W” y “x”? Son las iniciales de los **sistemas operativos** más usados: “L” de Linux (cualquiera de sus miles de variantes); “M”, de Mac; “W”, de Windows. Y la “x” se usa como un comodín, cuando se puede prescindir de un sistema operativo en particular y nos referimos a una instalación de Apache, MySQL y PHP genérica, en cualquier sistema operativo.

Esta sigla resume el entorno bajo el cual estamos usando a PHP. Así que, si están usando Windows y, a continuación, instalan Apache, MySQL y PHP, estarán trabajando bajo una plataforma WAMP (Windows con Apache, MySQL y PHP). Por el contrario, la mayoría de los *hostings* se basan en LAMP (Linux, Apache, MySQL y PHP).

El hosting

Eligiendo un buen hosting con PHP y MySQL

Sin lugar a duda, es imprescindible para probar nuestros desarrollos que dispongamos de un *hosting*, que puede funcionar bajo cualquiera de las plataformas recién mencionadas (aunque la más recomendable es **LAMP**, ya que es la que más potencia permite sacarle a PHP). ¡Atención!: esto no significa que nosotros debamos usar Linux en nuestra computadora, sino que el *hosting* usará ese sistema operativo. Nosotros simplemente nos conectaremos mediante algún programa de FTP para colocar en ese servidor nuestros archivos, y podemos hacer esto desde cualquier sistema operativo.

Para contratar algún servicio de *hosting*, debemos tener en cuenta que no son todos iguales, sino que existen distintas versiones de PHP y pueden tener instalada cualquiera de ellas. Lo ideal es conseguir *hostings* que posean la versión de PHP más actualizada que nos resulte posible: podemos consultar cuál es el número de la última versión de PHP si entramos a la Web oficial de PHP, en <http://www.php.net>

Además de buscar *hostings* con una versión actualizada de PHP, también debemos intentar que posean una versión lo más actualizada posible de MySQL, y del mismo modo que con PHP, el número de versión lo averiguaremos entrando a la Web oficial, en este caso, de MySQL: <http://www.mysql.com>

Hechas esas recomendaciones, el resto es sentido común: es mejor un *hosting* con soporte 24 horas, que podamos pagarlo en nuestra moneda local sin gastos de transferencia, y que no sea el de moda ni el más barato, porque suelen tener problemas frecuentemente. Suele ser útil que posean teléfono al que podamos llamar al costo de una llamada local para gestionar reclamos con más efectividad que por *e-mail* o ticket de soporte. Un *hosting* promedio, sin demasiados usuarios, suele ser mejor negocio que otro más barato pero saturado en su capacidad de dar soporte (más que un producto, el *hosting* es un servicio y es clave que tengamos acceso a las personas que nos podrán dar ayuda en caso de necesidad, de nada sirven esos soportes técnicos que repiten respuestas mecánicamente).

El servidor local para pruebas

Si bien siempre probaremos nuestros códigos en el *hosting* que nuestro cliente usará para su proyecto (para luego no tener sorpresas de configuración a la hora de dejar el sistema *on-line*), será muchísimo más práctico probar previamente nuestro código PHP localmente, en nuestra propia computadora, mientras

programamos, sin necesidad de esperar a transferir los archivos por FTP al *hosting* ante cada mínimo cambio que hagamos, ya que resulta muy molesto.

Para trabajar con un **servidor Web local** (dicho en palabras poco técnicas, una especie de simulador de *hosting*), tendremos que colocar nuestros archivos dentro de una **carpeta** en particular, que contendrá todos los archivos que programemos, tanto los ejercicios de este libro como nuestros propios proyectos profesionales. Y para que esos archivos funcionen, tendremos que mantener encendido un programa denominado servidor Web que, justamente, le “servirá” al navegador esos archivos ya procesados.

Ahora vamos a **descargar** e **instalar** ese software que nos permitirá montar nuestro propio servidor Web local.

Y, a continuación, veremos cómo acceder con el navegador a los archivos que serán servidos por ese software que instalemos.

Cómo descargar un servidor de pruebas

Aunque podríamos instalar todo el software necesario para programar en PHP y MySQL manualmente, es un trabajo bastante complejo y es probable cometer errores de configuración difíciles de solucionar sin ayuda de un Administrador de Sistemas. Por eso, es que existen muchos **instaladores automáticos** de todos los programas necesarios para probar código PHP en nuestra propia computadora: algunos de los nombres de estos instaladores son easyPHP, XAMPP, AppServ, etc. En este libro, utilizaremos el **XAMPP**, que es un instalador automático, que configura en instantes todos estos programas:

1. Un programa **servidor Web** llamado Apache,
2. El programa **intérprete del lenguaje PHP** propiamente dicho,
3. Un **programa gestor de bases de datos** denominado MySQL,
4. Una **interfaz visual** para interactuar con esas bases de datos, cuyo nombre es phpMyAdmin.

Este práctico paquete instalador “todo en uno”, se descarga gratuitamente de: [http:// www.apachefriends.org](http://www.apachefriends.org). Buscamos la versión correspondiente a nuestro sistema operativo (Windows, Linux, Mac, Solaris) y descargamos el instalador.

Cómo instalar el servidor de pruebas

Una vez terminada la descarga, haremos doble clic en el instalador que hemos descargado, y aceptaremos todo lo que nos pregunte, con total confianza (es software libre además de gratuito, así que nos permitirá utilizarlo sin limitaciones, de forma totalmente legal).

Una vez elegido el idioma de la instalación (en español), y manifestado nuestro acuerdo con la licencia de los productos, nos advertirá (en inglés) que no se recomienda utilizar este instalador en un *hosting* público real, por lo que seguimos adelante pulsando en **Siguiente** y, aquí, llegamos al momento crítico de la instalación: cambiaremos la **ruta de instalación** por defecto, por otra. Podemos ver que la ruta por defecto que figura en **Destination folder** es “C:\” así que la cambiaremos por esta:

C:\servidor\

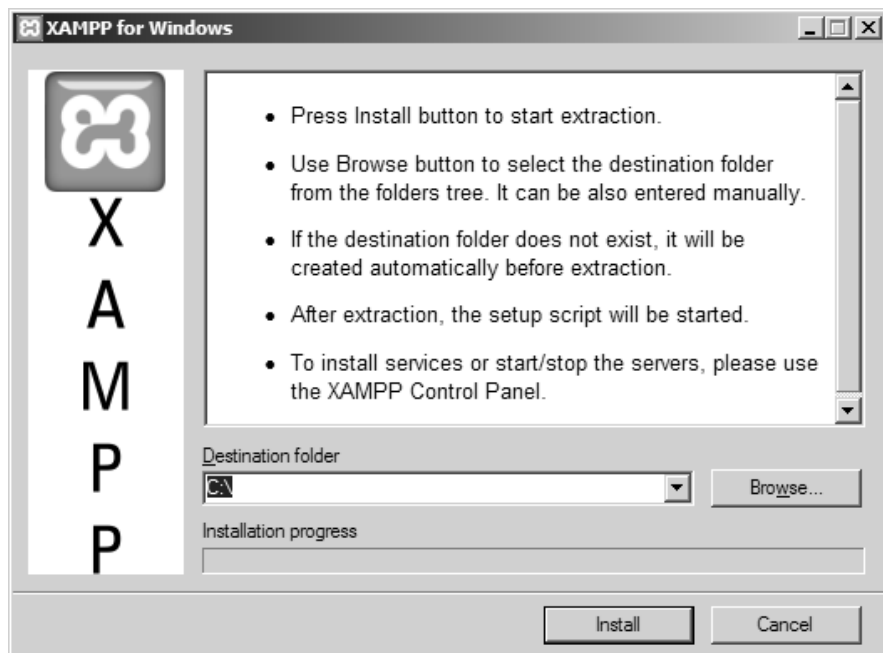


Figura. 2-1. Ruta por omisión.

Para cambiarla, simplemente **escribimos desde la última barra en adelante**, y solamente la palabra “servidor\”, para que quede de esta manera:

Desde ya que no es obligatorio utilizar exactamente esta misma ruta de instalación, el lector puede elegir libremente cualquier otra carpeta distinta si así lo prefiere: es solo para que coincida con lo que se mencionará en otras partes del libro.

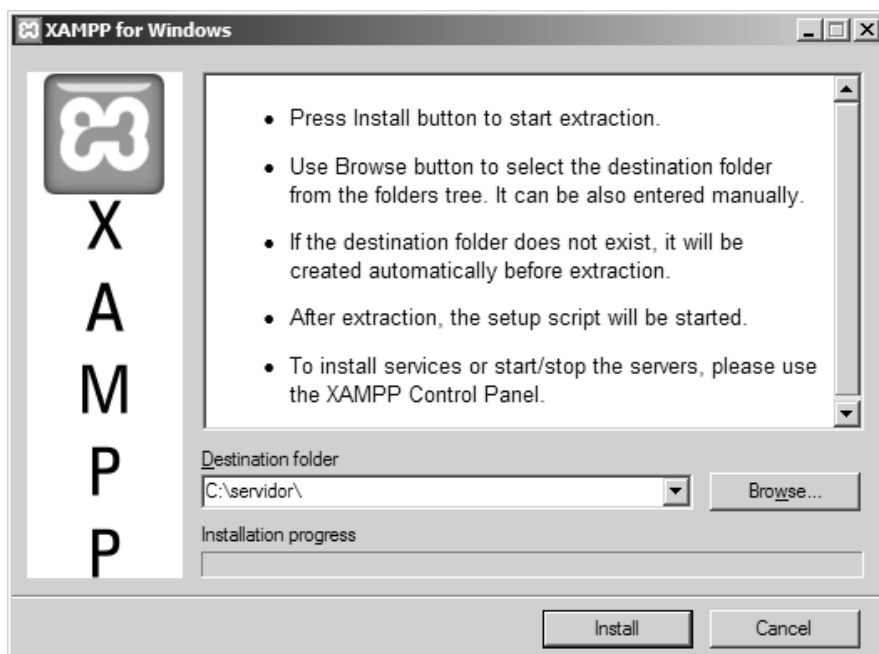


Figura 2-2. Ruta que utilizaremos en este libro.

Recomiendo que usemos **esa ruta**, ya que todas las explicaciones del libro están armadas tomando como base esa ruta de instalación.

A continuación, pulsamos el botón **Install**, y esperamos aproximadamente un minuto mientras extrae todos los paquetes necesarios.

Si al momento de leer este libro la versión de este instalador cambió o los pasos y pantallas son otros, esto no debe preocupar al lector; se debe simplemente seguir paso a paso las instrucciones que aparezcan en pantalla.

El único punto importante es tener conciencia de cuál es la carpeta en la que estamos instalando los servidores, el resto de configuraciones no es importante.

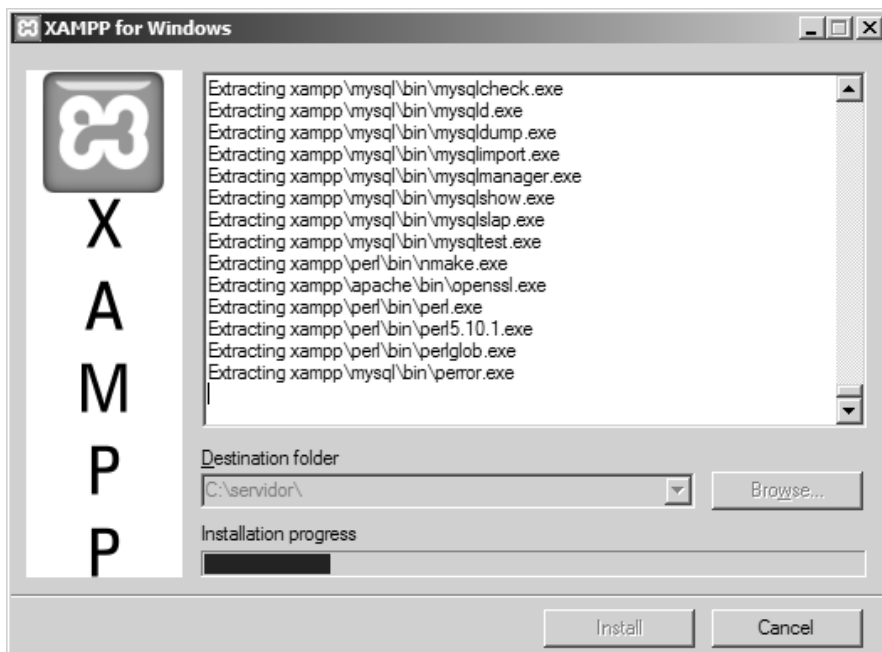


Figura 2-3. Pantalla que veremos mientras esperamos.

Por último, se abrirá una ventana de sistema que nos preguntará algunas cosas:

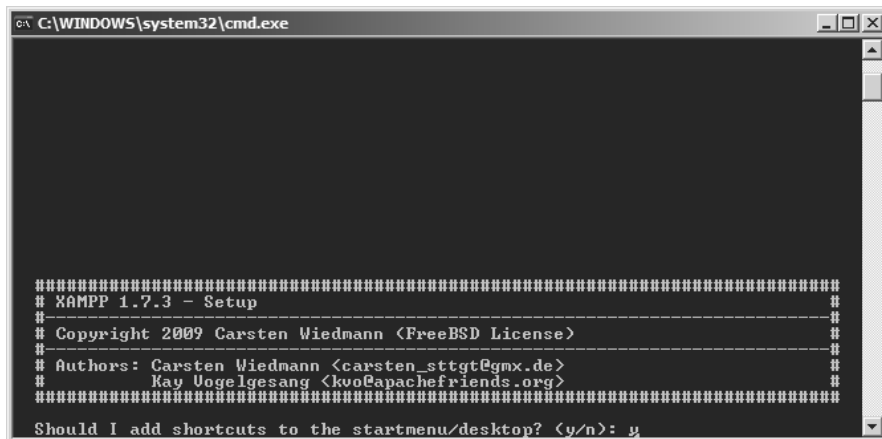
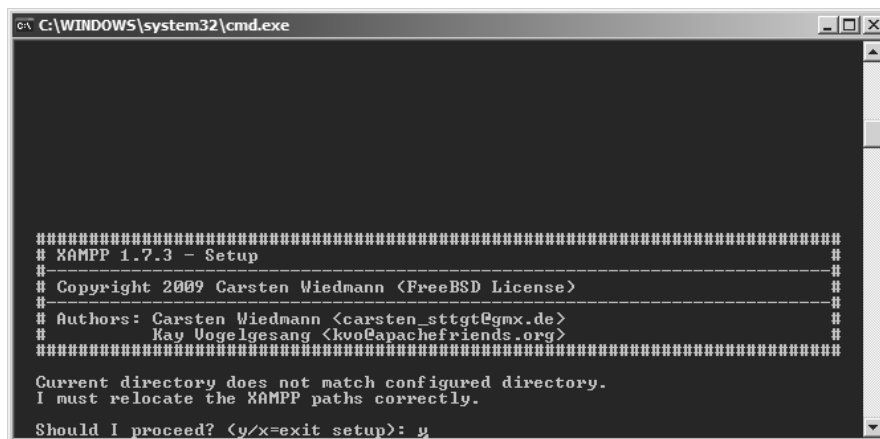


Figura 2-4. Accesos directos.

Nos está preguntando si agrega accesos directos al **menú Inicio** y al **Escritorio**, pulsamos una **y** para indicar que sí y, después, pulsamos **Enter**.

Luego, detectará que hemos cambiado la carpeta por omisión y nos pedirá que confirmemos la modificación de las rutas de todos los archivos de configuración (es fundamental que aquí respondamos que sí, pulsando **y** y, a continuación, **Enter**):

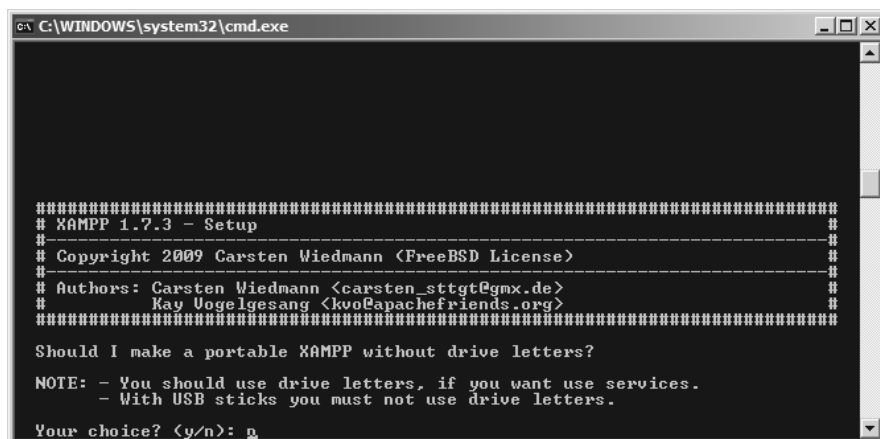


```
C:\WINDOWS\system32\cmd.exe

#####
# XAMPP 1.7.3 - Setup                                     #
# -----                                                #
# Copyright 2009 Carsten Wiedmann <FreeBSD License>      #
# -----                                                #
# Authors: Carsten Wiedmann <carsten_sttgt@gmx.de>       #
#          Kay Vogelgesang <kvo@apachefriends.org>        #
# -----                                                #
Current directory does not match configured directory.
I must relocate the XAMPP paths correctly.
Should I proceed? (y/x=exit setup): y
```

Figura 2-5. Confirmamos cambio de directorio de instalación.

Acto seguido, nos preguntará si nos interesaría crear una instalación portátil –para discos portátiles–, pero, como, por el momento, no es nuestra intención, pulsaremos la letra **n** y, luego, **Enter**:

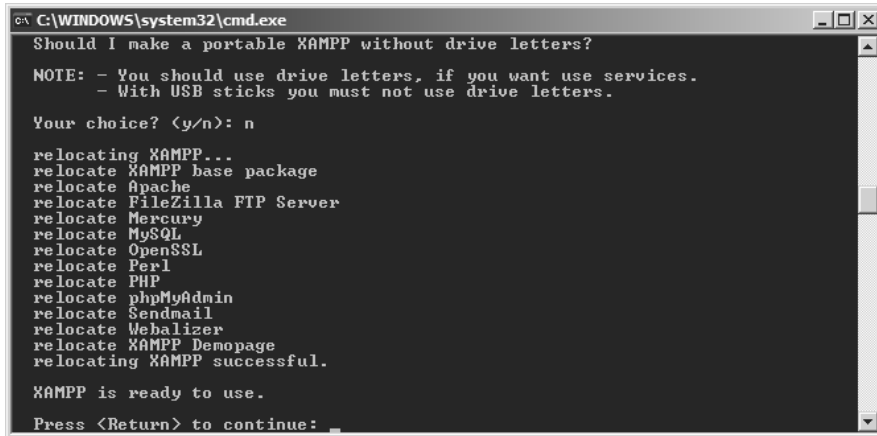


```
C:\WINDOWS\system32\cmd.exe

#####
# XAMPP 1.7.3 - Setup                                     #
# -----                                                #
# Copyright 2009 Carsten Wiedmann <FreeBSD License>      #
# -----                                                #
# Authors: Carsten Wiedmann <carsten_sttgt@gmx.de>       #
#          Kay Vogelgesang <kvo@apachefriends.org>        #
# -----                                                #
Should I make a portable XAMPP without drive letters?
NOTE: - You should use drive letters, if you want use services.
      - With USB sticks you must not use drive letters.
Your choice? (y/n): n
```

Figura 2-6. Manifestamos que ahora no estamos interesados en crear una instalación portable.

Ahora, realizará el cambio de configuración que habíamos autorizado dos pasos atrás, lo que demorará unas decenas de segundos, y nos indicará que pulsemos **Enter**:



```

C:\WINDOWS\system32\cmd.exe
Should I make a portable XAMPP without drive letters?
NOTE: - You should use drive letters, if you want use services.
      - With USB sticks you must not use drive letters.

Your choice? <y/n>: n

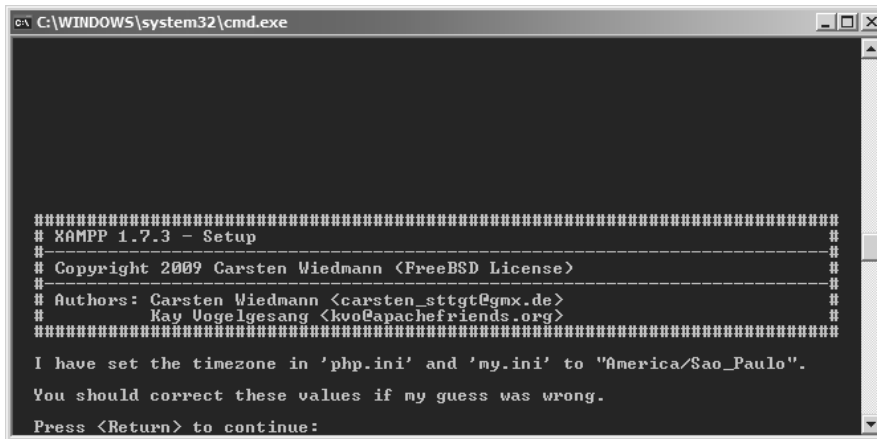
relocating XAMPP...
relocate XAMPP base package
relocate Apache
relocate FileZilla FTP Server
relocate Mercury
relocate MySQL
relocate OpenSSL
relocate Perl
relocate PHP
relocate phpMyAdmin
relocate Sendmail
relocate Webalizer
relocate XAMPP Demopage
relocating XAMPP successful.

XAMPP is ready to use.
Press <Return> to continue:

```

Figura 2-7. Final de la instalación.

Por último, configurará la zona horaria según lo detectado en el sistema operativo, y nos pedirá que pulsemos **Enter** otra vez:



```

C:\WINDOWS\system32\cmd.exe

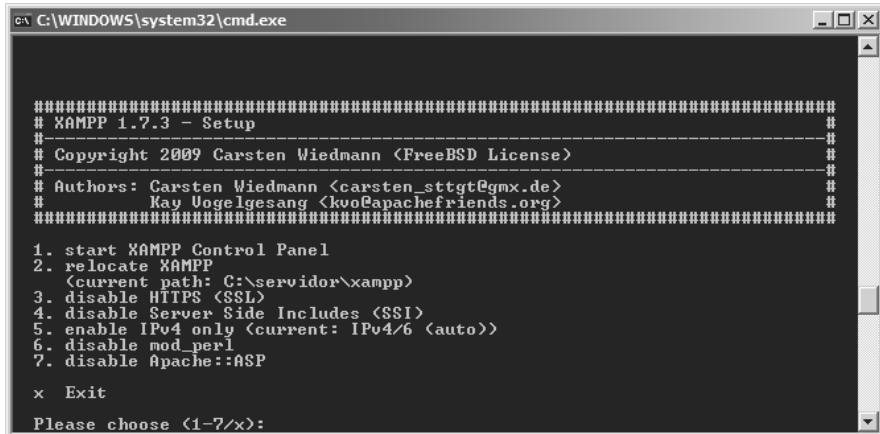
#####
# XAMPP 1.7.3 - Setup                                     #
# -----                                                #
# Copyright 2009 Carsten Wiedmann <FreeBSD License>      #
# -----                                                #
# Authors: Carsten Wiedmann <carsten_sttgt@gmxd.de>      #
#          Kay Vogelgesang <kvo@apachefriends.org>        #
#####

I have set the timezone in 'php.ini' and 'my.ini' to "America/Sao_Paulo".
You should correct these values if my guess was wrong.
Press <Return> to continue:

```

Figura 2-8. Autodetección de zona horaria.

Llegado este momento, nos muestra una pantalla con un menú de opciones:



```
C:\WINDOWS\system32\cmd.exe

#####
# XAMPP 1.7.3 - Setup                                     #
#####
# Copyright 2009 Carsten Wiedmann <FreeBSD License>      #
#####
# Authors: Carsten Wiedmann <carsten_stt@tgm.de>         #
#           Kay Vogelgesang <kvo@apache@friends.org>      #
#####

1. start XAMPP Control Panel
2. relocate XAMPP
   <current path: C:\servidor\xampp>
3. disable HTTPS <SSL>
4. disable Server Side Includes <SSI>
5. enable IPv4 only <current: IPv4/6 <auto>>
6. disable mod_perl
7. disable Apache::ASP

x Exit

Please choose (1-7/x):
```

Figura 2-9. Opciones para comenzar.

Podemos cerrar esta ventana de sistema escribiendo una **x** y pulsando **Enter**, porque ya tenemos todos los programas necesarios instalados.

Cómo encender y apagar el servidor de pruebas

1. Cómo encenderlo:

Para “encender” este software, debemos pulsar el ícono que dice **XAMPP Control Panel**, que quedó en nuestro Escritorio, o ingresar al menú:

Inicio -> Todos los programas -> XAMPP for Windows -> XAMPP Control Panel

Esto abrirá el siguiente panel que controla el encendido y apagado de todas las aplicaciones que fueron instaladas por el XAMPP. (ver Fig. 2-10)

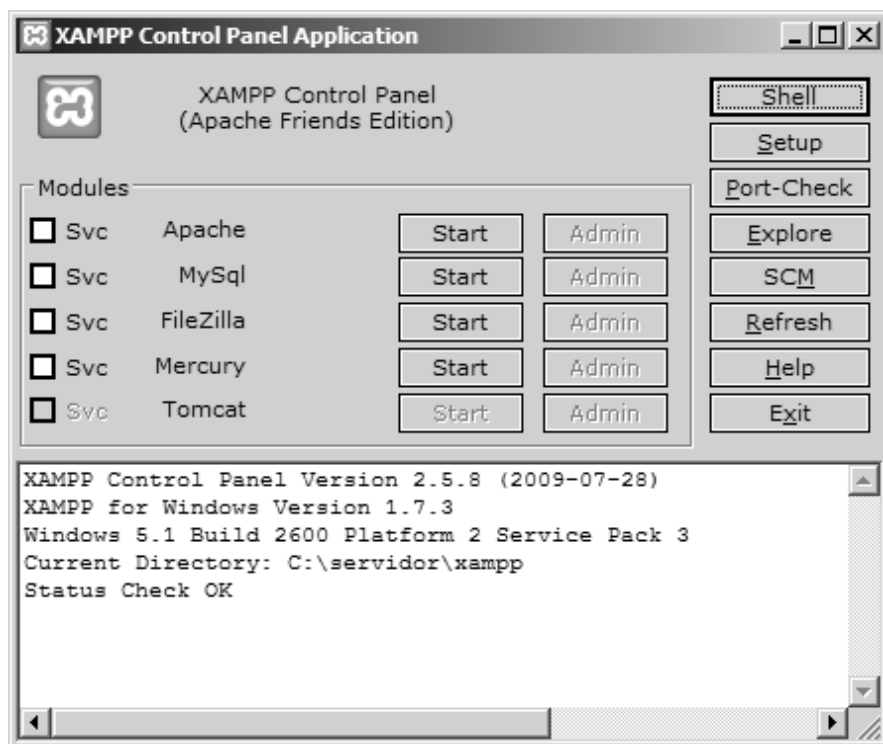


Figura 2-10. Panel de control de las aplicaciones instaladas por el XAMPP.

Para encender el servidor Web Apache, que es lo que precisamos por ahora para programar en PHP, pulsamos el botón **Start** que se encuentra a la derecha de la palabra **Apache**: sucederán varias cosas: la primera, ese botón ahora dirá **Stop**; la segunda, a la izquierda del botón que acabamos de apretar aparecerá la palabra **Running** sobre un fondo verde, que indica que el servidor Web Apache ya está encendido; además, en el recuadro blanco en el que aparecen los mensajes de texto, veremos que dice **Apache started**.

Cuando, más adelante en el proceso de aprendizaje, utilicemos el sistema gestor de bases de datos MySQL, lo encenderemos desde este mismo panel. Por ahora, solamente encenderemos el servidor web Apache, que incluye como uno de sus módulos al programa intérprete de lenguaje PHP.

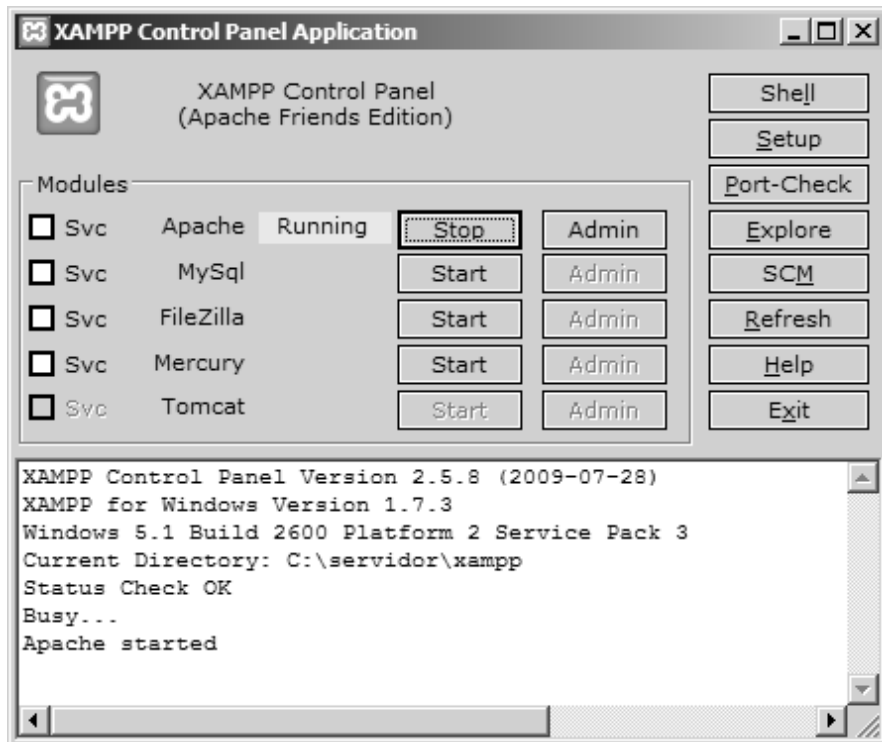


Figura 2-11. Panel de control con el servidor Web Apache encendido.

Si es la **primera vez** que lo encendemos, el *firewall* de Windows abrirá este cuadro de diálogo, que nos pedirá permiso para desbloquear el servidor Web que acabamos de encender. Para poder usar el servidor Web local, debemos pulsar en **Desbloquear**.

Según la versión del sistema operativo que estemos utilizando, las pantallas del firewall pueden ser diferentes. Asimismo, si tenemos instalado algún antivirus o programa que incluya protección a nivel firewall, deberemos autorizar la ejecución de nuestro servidor, o de lo contrario no podremos utilizarlo.



Figura 2-12. Alerta del Firewall de Windows, para que autorizemos el desbloqueo del servidor Web Apache.

Ahora, podremos minimizar o cerrar el Panel de Control de XAMPP y, aunque lo cerremos, los programas que hemos iniciado seguirán funcionando.

Nos daremos cuenta de que el Panel de Control está abierto, si miramos en la barra de tareas de Windows y encontramos el ícono de XAMPP:



Figura 2-13. Barra de tareas mostrando el ícono de XAMPP encendido.

Si le hacemos clic a ese ícono en la barra de tareas, se abrirá nuevamente el Panel de Control.

2. Cómo apagarlo:

Cuando terminemos de programar en PHP, podremos “apagar” el servidor Web, para eso, abriremos el Panel de Control de XAMPP, y pulsaremos el botón gris que dice **Stop**, a la derecha de **Apache** y de la palabra **Running** sobre fondo verde, y se apagará el servidor Web Apache en pocos segundos.

Cómo configurar el servidor de pruebas

Para programar localmente en PHP con mayor comodidad, es muy conveniente dedicar un minuto a configurar que el programa intérprete de PHP nos avise cada vez que cometamos un **error de sintaxis** (cosa que en un servidor Web de un