

USERS

ACTUALIZADO
A LA NUEVA
VERSIÓN:
PHP 6!

PHP AVANZADO

**DOMINE EL LENGUAJE MÁS
CONFiable Y ESTABLE**

WEB 2.0 Y AJAX A TRAVÉS DE PHP

DESARROLLO DE APLICACIONES CON CAKEPHP

GENERACIÓN Y TRATAMIENTO DE GRÁFICOS

SEGURIDAD EN APLICACIONES WEB

USO DE EXTENSIONES PEAR

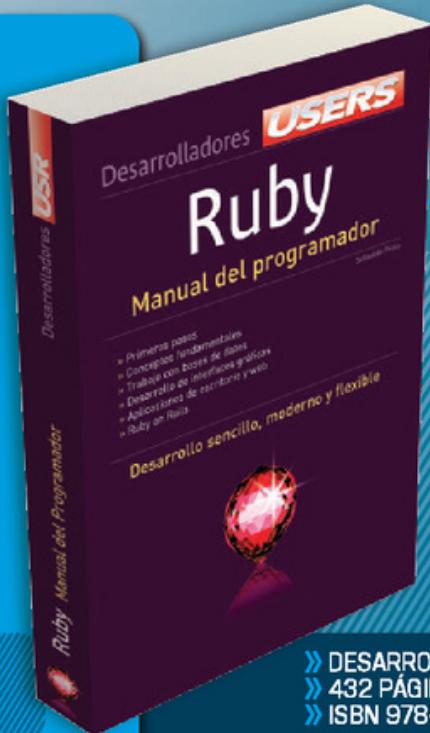
por Francisco Minera



MANUALES USERS MANUALES USERS MANUALES USERS MANUALES USERS MANUALES

PROGRAMACIÓN DE SITIOS WEB PROFESIONALES

CONÉCTESE CON LOS MEJORES LIBROS DE COMPUTACIÓN



DESARROLLO DE
ÚLTIMA GENERACIÓN
CON UN LENGUAJE
LIBRE Y FLEXIBLE

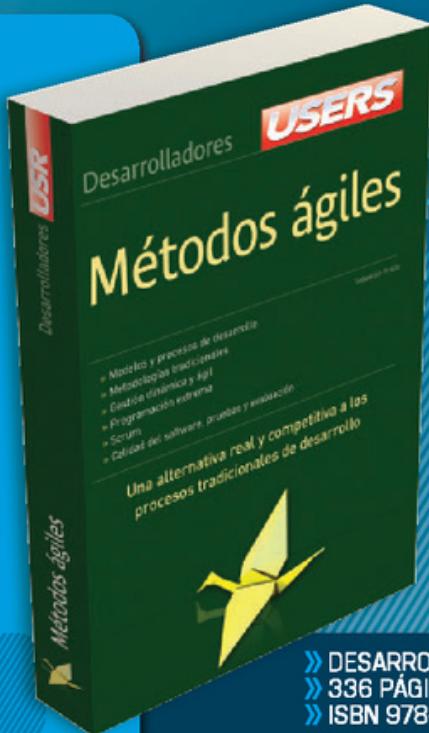
- » DESARROLLO
- » 432 PÁGINAS
- » ISBN 978-987-1347-67-4

LLEGAMOS A TODO EL MUNDO
VÍA **DODA*** Y **PHL****

● usershop.redusers.com
✉ usershop@redusers.com



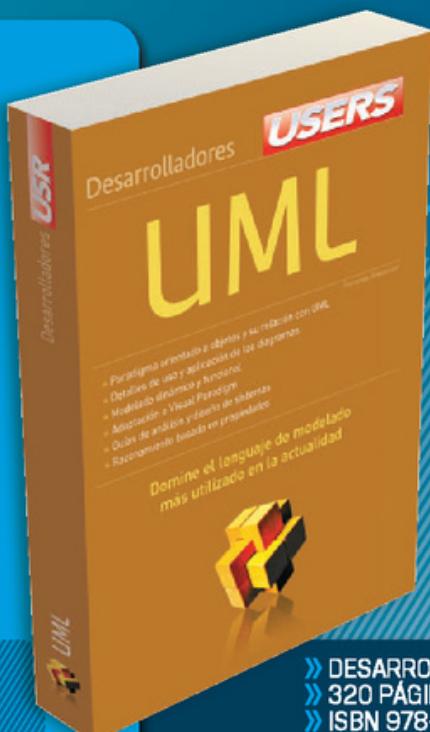
* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA



UNA ALTERNATIVA
A LOS MÉTODOS
TRADICIONALES
DE DESARROLLO

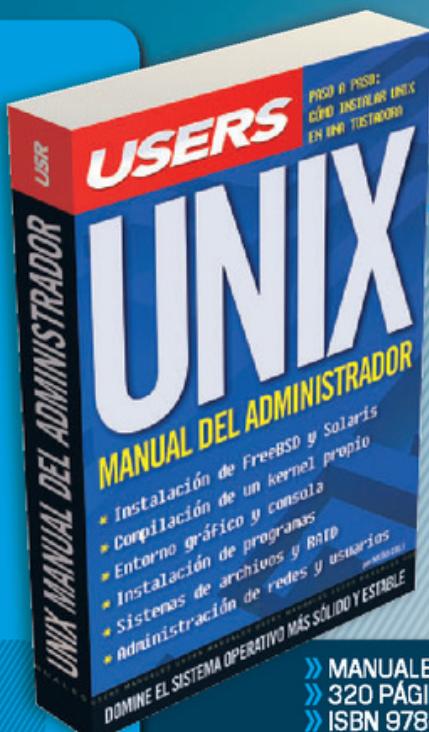


- » DESARROLLO
- » 336 PÁGINAS
- » ISBN 978-987-1347-97-1



COMIENCE A
DESARROLLAR
DESDE LA
PERSPECTIVA
DE LOS MODELOS

- » DESARROLLO
- » 320 PÁGINAS
- » ISBN 978-987-1347-95-7



DOMINE EL SISTEMA
OPERATIVO MÁS
SÓLIDO, CONFiable
Y ESTABLE

- » Instalación de FreeBSD y Solaris
- » Compilación de un kernel propio
- » Entorno gráfico y consola
- » Instalación de programas
- » Sistemas de archivos y RAID
- » Administración de redes y usuarios

- » MANUALES USERS
- » 320 PÁGINAS
- » ISBN 978-987-1347-94-0

PHP AVANZADO

DOMINE EL LENGUAJE MÁS CONFIABLE Y ESTABLE

por Francisco José Minera

RedUSERS



TÍTULO: PHP AVANZADO

AUTOR: Francisco Minera

COLECCIÓN: Manuales USERS

FORMATO: 17 x 24 cm

PÁGINAS: 400

Copyright © MMX. Es una publicación de Fox Andina en coedición con Gradi S.A. Hecho el depósito que marca la ley 11723. Todos los derechos reservados. No se permite la reproducción parcial o total, el almacenamiento, el alquiler, la transmisión o la transformación de este libro, en cualquier forma o por cualquier medio, sea electrónico o mecánico, mediante fotocopias, digitalización u otros métodos, sin el permiso previo y escrito del editor. Su infracción está penada por las leyes 11723 y 25446. La editorial no asume responsabilidad alguna por cualquier consecuencia derivada de la fabricación, funcionamiento y/o utilización de los servicios y productos que se describen y/o analizan. Todas las marcas mencionadas en este libro son propiedad exclusiva de sus respectivos dueños. Impreso en Argentina. Libro de edición argentina. Primera impresión realizada en Sevagraf, Costa Rica 5226, Grand Bourg, Malvinas Argentinas, Pcia. de Buenos Aires en XII, MMX.

ISBN 978-987-1773-07-7

Minera, Francisco

PHP avanzado. - 1a ed. - Buenos Aires : Fox Andina; Banfield - Lomas de Zamora: Gradi, 2011.
v. 202, 400 p. ; 24x17 cm. - (Manual Users)

ISBN 978-987-1773-07-7

1. Informática. I. Título

CDD 005.3



LÉALO ANTES GRATIS

EN NUESTRO SITIO PUEDE OBTENER, DE FORMA GRATUITA, UN CAPÍTULO DE CADA UNO DE LOS LIBROS

RedUSERS
COMUNIDAD DE TECNOLOGIA

 **redusers.com**

Nuestros libros incluyen guías visuales, explicaciones paso a paso, recuadros complementarios, ejercicios, glosarios, atajos de teclado y todos los elementos necesarios para asegurar un aprendizaje exitoso y estar conectado con el mundo de la tecnología.



LLEGAMOS A TODO EL MUNDO VÍA  * Y  **

* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

 usershop.redusers.com //  usershop@redusers.com

Francisco José Minera



Es Analista de Sistemas y Programador Superior. Desde 2002, se ocupa del desarrollo de sistemas y plataformas para pequeñas y medianas empresas en diversos lenguajes y arquitecturas.

Su vasta experiencia es volcada en sus libros previos: *PHP* y *MySQL*, *Proyectos con PHP, XML, PHP 5 y Ajax*, todos publicados por esta misma editorial. Puede contactar con el autor para enviar sugerencias o comentarios a su dirección de correo electrónico: francisco.minera@gmail.com

Dedicatoria

Este libro está dedicado a mis familiares, en especial a María Eugenia, y a todos los que hicieron posible su publicación.

Prólogo

Uno de los puntos que motivaron la reedición de este libro fue el hecho de actualizar el material incluido en el mismo, y luego de haber realizado el trabajo podemos sacar en limpio algunas de las siguientes ideas, algunas referidas a proyectos particulares y otras al lenguaje en general:

- **Evolución.** En las mayoría de los proyectos se nota claramente la tendencia hacia mejorar la estructura interna de las aplicaciones, manteniendo el código actualizado según las ultimas técnicas de programación. Esto repercute directamente en la calidad final de los scripts, lo que permite a su vez que nuestras aplicaciones se vean beneficiadas. En relación a este punto, otra consecuencia positiva es que el ver tanta motivación y constante búsqueda de superación nos anima a aplicar esos mismos conceptos en nuestros propios emprendimientos.
- **Nuevas versiones.** PHP distribuye nuevas versiones de manera constante, lo que nos habla del interés que se tiene en mantener el estatus logrado desde hace ya varios años. Un punto a tener en cuenta es que las nuevas versiones se complementan con las anteriores, lo que implica que en la mayoría de los casos no tendremos que modificar nuestro código de manera urgente.
- **Código abierto.** El lenguaje de programación PHP mantiene su gran popularidad -e incluso la amplia frente a sus competidores- y a la vez continua con sus principios fundamentales: la enorme mayoría de las aplicaciones son de libre descarga y uso, y su código fuente puede ser revisado, estudiado, modificado, corregido o ampliado según lo creamos conveniente.
- **Nuevas opciones.** Algo digno de ser mencionado es la expansión permanente en cuanto a los alcances de las aplicaciones: en muchos casos se da que van mas allá de su propósito inicial, empujadas por una gran masa de usuarios que con sus ideas, aportes o pedidos motivan a los desarrolladores a añadir nuevas funcionalidades, sin duda, este es el caso de PHP.
- **Comunidad.** Es importante saber que se ve una permanente participación por parte de los usuarios a la hora de opinar, colaborar e incluso aportar horas de trabajo para afianzar y mejorar proyectos iniciados por otras personas. Este flujo de actividad hace que los proyectos tengan vida propia, logrando mantenerse a la vanguardia en cuanto a las funcionalidades ofrecidas.

Debemos saber que todos estos conceptos hablan sin dudas de la filosofía que este lenguaje ha logrado afianzar durante tanto tiempo en el mercado: funcionalidad, autosuperación, revisión permanente, y la búsqueda de nuevos caminos que extiendan las posibilidades ya existentes.

EL LIBRO DE UN VISTAZO

El desarrollo de aplicaciones utilizando PHP se ha visto modificado en los últimos tiempos, y este libro apunta a afianzar los conocimientos indispensables que nos permitirán generar aplicaciones profesionales de nivel. Cada capítulo está dedicado a una técnica específica e ilustrado mediante ejemplos prácticos listos para implementar.

Capítulo 1

PLANEAMIENTO Y CONTROL DE PROYECTOS

En este primer capítulo, veremos las bases esenciales para generar aplicaciones libres de error y mantenibles a lo largo del tiempo, todo a través de las funcionalidades provistas por PHP. Incluiremos

PHPDocumentor, una de las alternativas más populares, y todo lo referido al control de errores, utilizando las configuraciones disponibles en el lenguaje.

Capítulo 2

TEMPLATES

La separación entre la parte de diseño de un sitio en relación con la parte lógica del programa es indispensable a la hora de desarrollar proyectos profesionales.

Aquí haremos un repaso por una de las alternativas más populares diseñadas para tal fin: Sigma, la clásica herramienta utilizada en el mundo PEAR, y Smarty, una opción profesional para desarrollos de todo tipo.

Capítulo 3

WEB 2.0: AJAX

PHP da una respuesta a cada una de las demandas de sus usuarios, y el fenómeno Ajax no podía ser la excepción. En este

capítulo, conoceremos Sajax, Xajax y jPOP, algunas de las opciones para aquellos desarrolladores que pretendan incluir en sus aplicaciones todas las funcionalidades disponibles de este modelo, en el momento de implementarlo.

Capítulo 4

RAD EN PHP

El desarrollo rápido de aplicaciones es ya una realidad para todo tipo de programadores gracias a herramientas como CakePHP tratada en este capítulo a través de ejemplos prácticos de sus diversos usos. Se incluye también la vinculación de CakePHP con bases de datos y las opciones disponibles para relacionar Modelos dentro de una aplicación de tipo MVC con PHP.

Capítulo 5

SERVICIOS WEB

PHP nos ofrece múltiples alternativas para comunicar aplicaciones, y en esta sección haremos un recorrido por las extensiones del lenguaje (incluyendo SOAP) y las clases PEAR disponibles. Ilustraremos su uso mediante ejemplos prácticos, listos para ser implementados y poder acceder a los servicios gratuitos ofrecidos por las compañías especializadas.

Capítulo 6**ORIENTACIÓN A OBJETOS**

En las últimas versiones del lenguaje, uno de los mayores avances fue que se actualizó casi en forma total la implementación de la POO en PHP. En este capítulo, haremos énfasis en las novedades disponibles, algo que nos ayudará a reformular nuestras aplicaciones para mantenerlas compatibles con las versiones más actuales de PHP.

Capítulo 8**PEAR**

El repositorio oficial de clases nos permite extender la potencia del lenguaje. Por eso, en este capítulo, explicamos los pasos para instalar, configurar y utilizar PEAR, más algunas de las opciones disponibles: MDB2, la capa de abstracción para acceder a bases de datos; Spreadsheet_Excel_Writer, y también el popular HTML_QuickForm.

Capítulo 7**SEGURIDAD EN APLICACIONES WEB**

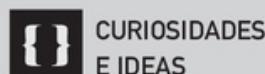
En esta sección, enumeramos los conceptos más importantes que nos permitirán desarrollar aplicaciones web seguras y profesionales, todos ellos ilustrados a través de ejemplos prácticos de utilización. Veremos temas como validación de datos externos, formularios web, bases de datos, y servicios web, entre otras posibilidades. Se incluye una referencia de las funciones más importantes.

Capítulo 9**GENERACIÓN Y TRATAMIENTO DE GRÁFICOS**

En la actualidad, la inclusión de elementos gráficos e imágenes forma parte de la gran mayoría de los desarrollos en los cuales podemos trabajar, y PHP nos provee la funcionalidad necesaria para manipular esta clase de archivos. En este capítulo veremos en particular cómo generar archivos estadísticos y también cómo debemos proceder a tratar imágenes ya existentes.

**INFORMACIÓN COMPLEMENTARIA**

A lo largo de este manual encontrará una serie de recuadros que le brindarán información complementaria: curiosidades, trucos, ideas y consejos sobre los temas tratados. Cada recuadro está identificado con uno de los siguientes iconos:



CURIOSIDADES
E IDEAS



ATENCIÓN



DATOS ÚTILES
Y NOVEDADES



SITIOS WEB



Desarrollos temáticos en profundidad

Libros.

Coleccionables.

Cursos intensivos con multimedia



Capacitación dinámica

Revistas.

Sitios Web.

Noticias al día, downloads, comunidad



Información actualizada al instante

Newsletters.

La red de productos sobre tecnología más importante del mundo de habla hispana.



redusers.com

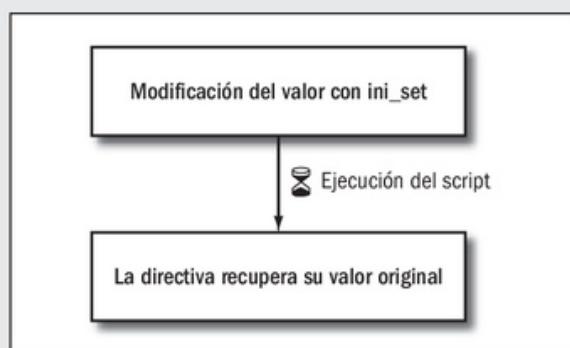
CONTENIDO

Sobre el autor	4
Prólogo	5
El libro de un vistazo	6
Introducción	12

Capítulo 1

PLANEAMIENTO Y CONTROL DE PROYECTOS

Ciclo de una aplicación	14
Documentación de proyectos	15
Manejo y control de errores	25
Directivas de configuración	25
Funciones	29
Bases de datos	31
Excepciones	34
Resumen	37
Actividades	38



Capítulo 2

TEMPLATES

Introducción	40
Sigma	40
Templates preparados	43
Variables	43
Bloques	45
Inclusión de archivos externos	50
Funciones	52
Otros métodos disponibles	53
Ejemplo completo de uso	53
Smarty	61

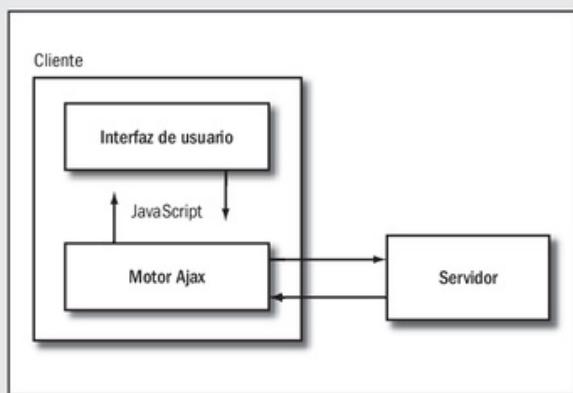
Modos de trabajo	61
Variables	63
Comentarios	66
Archivos de configuración	66
Propiedades y métodos del objeto Smarty	70
Operadores	71
Condicionales	72
Ciclos	72
Inclusión de archivos externos	74
Funciones	74
Caché	82
Objetos	83
Ejemplo completo de uso	84
Resumen	91
Actividades	92



Capítulo 3

WEB 2.0: AJAX

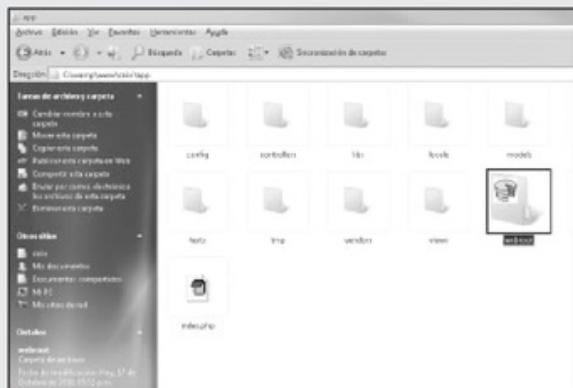
Origen y características	94
XML	100
JSON	102
Implementaciones en PHP	105
Sajax	106
Xajax	112
jPOP	122
Resumen	131
Actividades	132



Capítulo 4

RAD EN PHP

Características	134
CakePHP	135
Modelos	138
ABMC dinámicos	140
Relaciones entre modelos	144
Generadores	147
Templates	148
Ejemplo práctico de aplicación	150
Resumen	165
Actividades	166

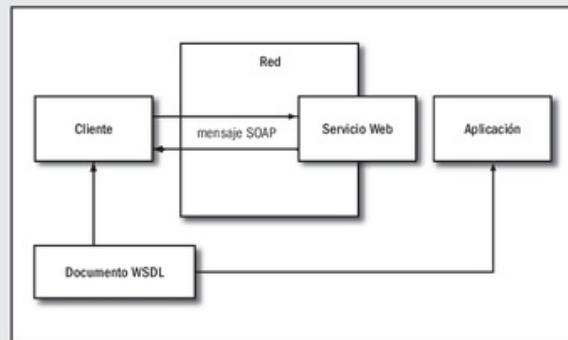


Capítulo 5

SERVICIOS WEB

Historia	168
Capas	169
Protocolos	171
XML-RPC	171
SOAP	172
WSDL	174

UDDI	176
Servicios web en PHP	177
Extensões disponibles	177
PEAR	181
Resumen	195
Actividades	196



Capítulo 6

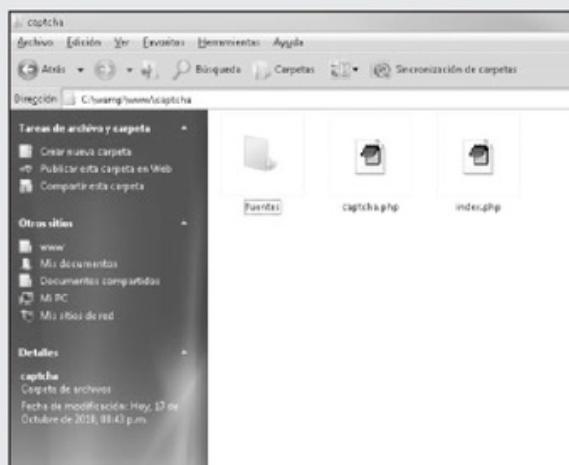
ORIENTACIÓN A OBJETOS

Antecedentes	198
Clases	199
Tipos de acceso a variables y funciones	200
Constantes	203
Constructores y destructores de clases	203
Subclases y forma de acceso	206
Captura de errores en clases, métodos y propiedades	207
Implementación de la herencia	209
Clases abstractas	213
Propiedades y métodos estáticos	215
Referenciación de objetos	218
Objetos como tipo de dato	221
Resumen	223
Actividades	224



Capítulo 7

SEGURIDAD EN APLICACIONES WEB	
Introducción	226
Filtrado de datos de entrada y salida	227
Formularios	230
CAPTCHA	234
Almacenar información en base de datos	241
El caso de MySQL	251
Seguridad en bases de datos	252
Inclusión de archivos	255
Ajax	256
Línea de comandos	257
Register Globals	258
Modo seguro en PHP	260
Manipulación del archivo php.ini	262
Seguridad en PHP 6	263
Resumen	263
Actividades	264

**Capítulo 8**

PEAR	
Instalación	266
Opciones disponibles	270
Utilización	272
Errores	273
Implementaciones	273
Acceso a datos con MDB2	273
Hojas de cálculo con Spreadsheet_Excel_Writer	290

Formularios con HTML_QuickForm	301
Agregar elementos	302
Validar formularios	305
Enviar datos	310
Salida sin tablas	311
Resumen	313
Actividades	314

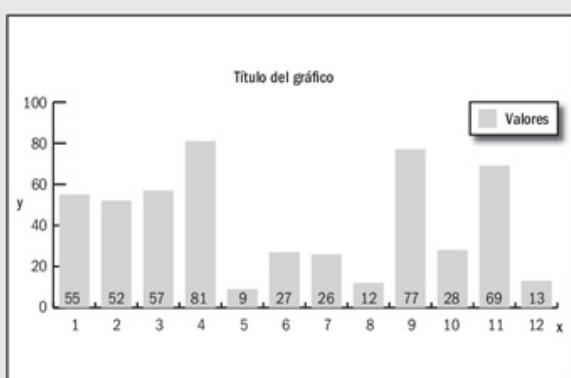
```
C:\WINDOWS\system32\cmd.exe
C:\wamp\php>pear list
INSTALLED PACKAGES, CHANNEL PEAR.PHP.NET:
-----
PACKAGE           VERSION   STATE
Archive_Tar       1.3.2    stable
Cache_Lite        1.7.8    stable
Config            1.10.11   stable
Console_Getopt   1.2.3    stable
File_SearchReplace 1.1.2    stable
HTML_Template_IT 1.3.0    stable
Image_Text        0.6.0beta beta
MDB2              2.4.1    stable
MDB2_Driver_mysql 1.4.1    stable
MIME_Type         1.2.0    stable
OLE               0.5      beta
PEAR              1.6.2    stable
PEAR_Frontend_Web 0.7.4    beta
Pager             2.4.8    stable
Spreadsheet_Excel_Writer 0.9.1    beta
Structures_Graph 1.0.2    stable
Text_Capitalize   0.4.0    stable
```

Capítulo 9**GENERACIÓN Y TRATAMIENTO**

DE GRÁFICOS	
La librería GD	316
PHP Thumbnailer	319
phpThumb	338
JpGraph	349
PEAR	378
Imagick	386
Resumen	389
Actividades	390

Servicios al lector

Índice temático	392
-----------------	-----



INTRODUCCIÓN

Este libro apunta a todos aquellos que, conociendo los fundamentos más básicos del lenguaje, quieren explorar sus posibilidades a un nivel superior, ampliando y mejorando sus propias aplicaciones. Tal vez sea necesaria una mínima experiencia o conocimiento del lenguaje HTML, pero no es algo determinante. Cabe destacar que uno de los objetivos de este libro es que sirva al lector como punto de partida para llevar adelante sus propios desarrollos y a la vez profundizar en los temas más cercanos a su interés, ya sea tanto a través de otros libros como por intermedio de tutoriales o revistas especializadas.

Sin dudas los temas aquí tratados son de propósito general, lo que significa que podremos aplicar los conceptos y las herramientas utilizadas en distintas clases de proyectos. Avanzando incluso un poco más, podemos decir que este libro nos ayudará no solo a utilizar soluciones ya desarrolladas sino también a crear las nuestras propias. Todas las herramientas utilizadas en esta obra son gratuitas, por lo cual podremos incluirlas sin problemas en nuestros sitios. Además existen en Internet numerosos informes acerca de cómo utilizarlas aplicando conceptos avanzados para sacarles el mayor provecho posible.

Cada capítulo de esta obra puede leerse en forma individual, esto quiere decir que si, por ejemplo, el lector no considera interesante o de utilidad una determinada sección, simplemente puede pasarla por alto y de esta forma continuar con la de su interés particular. La intención de esto es brindar fluidez y naturalidad al proceso de aprendizaje, y a la vez poder satisfacer las necesidades de quien accede a esta obra de manera directa y sin preámbulos.

Hay muchos factores que intervienen a la hora de desarrollar de manera exitosa una aplicación, y uno de ellos es saber con precisión cuál es la mejor opción disponible para llevar a cabo una determinada tarea. Ese es uno de los objetivos del libro: tratar de brindar al lector opciones para que este pueda aplicar su sentido común y así lograr el resultado esperado.

Luego de haber completado la lectura del libro, tomaremos plena conciencia de que podemos sin dudas ilusionarnos con respecto a los próximos años en cuanto al lenguaje en sí y a las aplicaciones basadas en el mismo. El futuro es tan o más prometedor que el presente, y la gran masa de desarrolladores que utilizan PHP avalan con su confianza a este lenguaje que tantas satisfacciones ha brindado tanto a usuarios como a clientes.

Planeamiento y control de proyectos

Al comenzar a desarrollar un sistema, aparecen cuestiones específicas referidas a qué se debe hacer, y otras generales que apuntan a cómo llevarlas a cabo. En este capítulo, haremos una serie de recomendaciones para tener en cuenta a la hora de programar nuestras aplicaciones.

Ciclo de una aplicación	14
Documentación de proyectos	15
Manejo y control de errores	25
Directivas de configuración	25
Funciones	29
Bases de datos	31
Excepciones	34
Resumen	37
Actividades	38

CICLO DE UNA APLICACIÓN

Un código escrito por un desarrollador, probablemente, necesite ser modificado a lo largo del tiempo ya sea por su autor o por terceros; lo que implica que debe respetar reglas implícitas, aceptadas por todos. Éstas sirven para mantener el código escrito en buen estado, apto para que otros lo amplíen, lo alteren o, simplemente, lo interpreten sin mayores complicaciones.

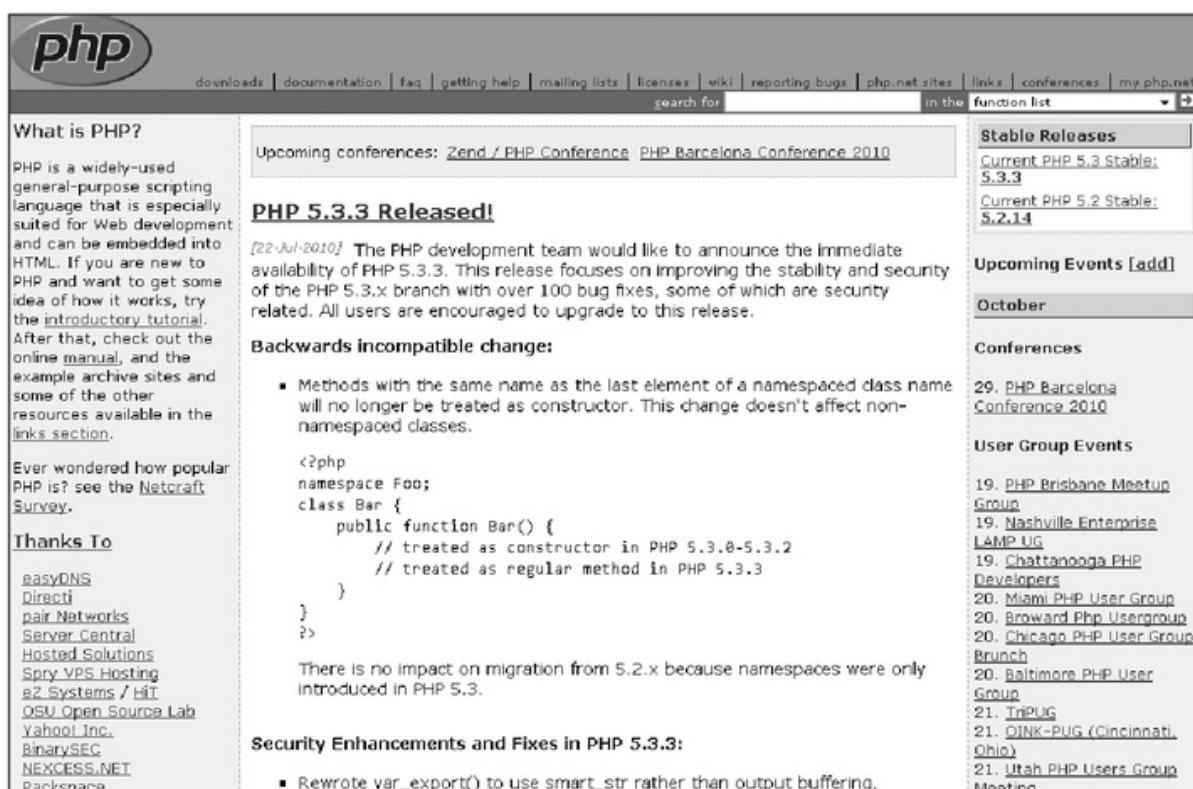


Figura 1. PHP incluye en sus últimas versiones avances considerables y decisivos con relación a las anteriores.

Hay dos puntos fundamentales que debemos tener en cuenta ante el desafío de comenzar el desarrollo de un nuevo proyecto: la forma en la que escribimos el código y la documentación de éste.

Sin importar el estilo de programación que utilicemos (orientado a objetos, procedural, etc), en principio, deberíamos redactar código limpio y apto para la comprensión de futuros lectores, incluso la de nosotros mismos. Estas reglas no son estrictas, y entre ellas podemos citar:

- Uso de identación.
- Ancho razonable de las líneas de código.
- Utilización de espacios cuando corresponda.
- Continuación del estilo inicial de programación.

Normalmente, la documentación de código se divide en dos partes: la inclusión de comentarios dentro de los archivos que forman parte de nuestra aplicación y la escritura de manuales o de guías por fuera del código.

El uso de una u otra (o de ambas) dependerá de la magnitud y de las necesidades de nuestros proyectos. Lo que sucede habitualmente es que, como desarrolladores, por lo menos al inicio, comprendemos el código escrito por nosotros mismos y no consideramos necesario incluir ninguna clase de comentarios ni anotaciones para futuras modificaciones de terceros. Mala idea.

DOCUMENTACIÓN DE PROYECTOS

PHPDocumentor (www.phpdoc.org) es una popular alternativa utilizada por los desarrolladores a la hora de agregar documentación de manera automática a nuestros sistemas, generándola desde el propio código fuente de la aplicación. Es un proyecto mantenido por **PEAR** (veremos más información en el **Capítulo 8**) y sucede a otra herramienta similar llamada **PHPDoc**. Podemos descargar PHP Documentor desde el sitio oficial de PEAR, en la dirección <http://pear.php.net/package/PhpDocumentor/download> o desde el sitio web de Source Forge, que se encuentra en la dirección <http://sourceforge.net/projects/phpdocu>.

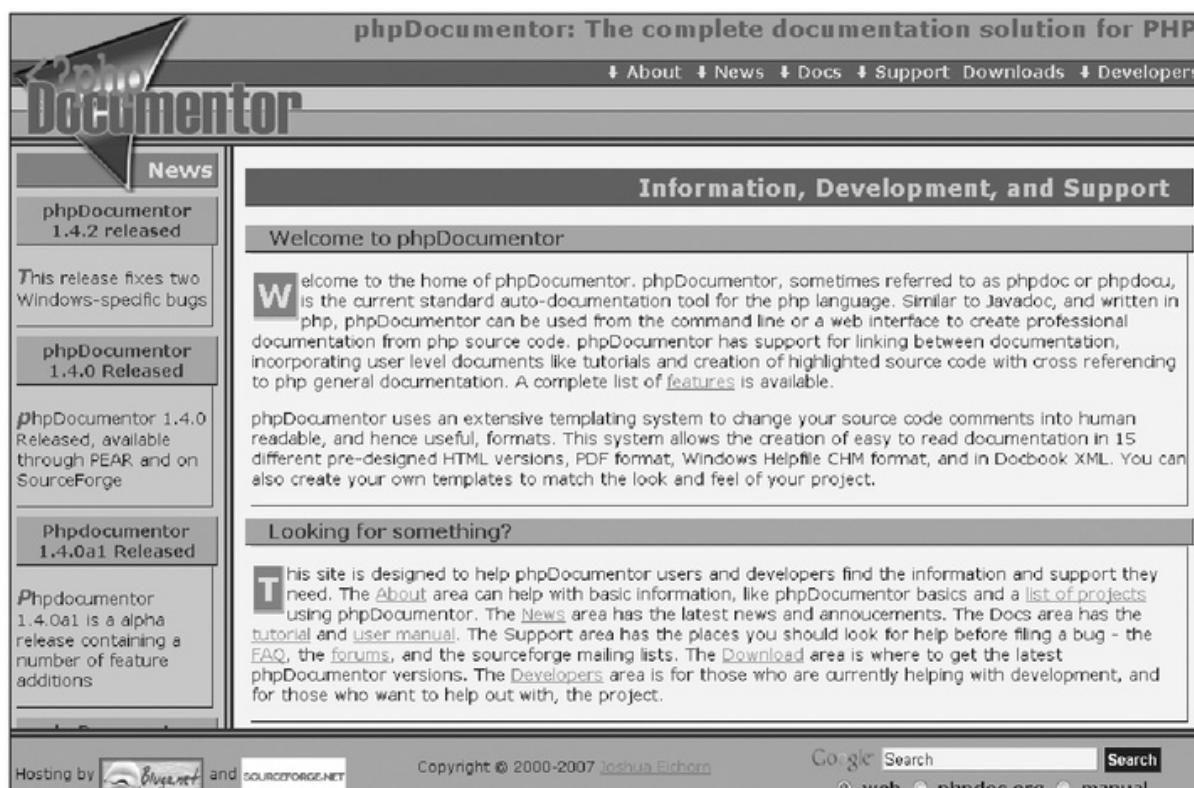


Figura 2. La documentación de aplicaciones juega un rol importante en el desarrollo en el ámbito profesional.

Esta clase de herramientas están disponibles para otros lenguajes como Perl (**POD**), Java (**JavaDoc**), o Python (**Pydoc**), entre otras, y nos permiten mantener un desarrollo listo para ser interpretado por los demás y actualizable a lo largo del tiempo.

En proyectos medianos y grandes, muchos desarrolladores participan aportando módulos que se acoplan a un sistema, lo cual les permite desligarse de las tareas ajena y focalizarse en las suyas específicas. Por diversas razones, cada grupo es responsable de su parte. Esta responsabilidad incluye tener la precaución de mantener el código en buena forma, lo que implica hacerlo entendible, en la medida de lo posible, a otros desarrolladores que heredarán el trabajo en el futuro. Otro caso de uso consiste en aquellos desarrollos que se liberan o se venden a usuarios o compañías y que, como valor agregado al producto, adosan la documentación pertinente.



Figura 3. *PHPDocumentor es un proyecto libre que podemos descargar desde el sitio oficial de PEAR. En el manual, se incluyen referencias de todas las funcionalidades disponibles.*

III CARACTERISTICA

Es importante saber que PHP es un lenguaje de programación caracterizado por ser bastante permisivo con relación a otros, en cuanto a la escritura de código, por lo que respetar las reglas mínimas para lograr desarrollos claros y bien estructurados es fundamental y recae casi exclusivamente en manos de cada programador, por esta razón debemos tener mucho cuidado.

La forma de trabajar con PHPDocumentor no difiere de otras alternativas similares: se encarga de leer el código fuente de la aplicación y, sobre la base de comentarios y declaraciones, genera la documentación apropiada de forma automática. Es capaz de generar archivos de ayuda en los siguientes formatos: **XML Docbook**, **HTML**, **PDF**, y **CHM** (generando un archivo **HHP**).

Para comenzar a utilizar esta herramienta, debemos respetar una determinada sintaxis. Los comentarios **DocBlock** (originalmente introducidos en el lenguaje C) serán luego tomados y procesados por PHPDocumentor para generar la documentación:

```
/**  
 * Esto es un comentario  
 */
```

Debemos agregar estos comentarios antes de las declaraciones que queramos incluir en la documentación, por ejemplo:

```
<?php  
  
/**  
 * Devuelve el cuadrado de un numero  
 *  
 * La funcion cuadrado recibe como argumento  
 * un numero y devuelve su cuadrado.  
 *  
 */  
function cuadrado($n)  
{  
    return $n*$n;  
}  
  
?>
```

Dentro de cada uno, podemos observar que contamos con dos tipos de descripciones: una corta (máximo de tres líneas) y una larga (sin límite). Deben estar separadas por una línea en blanco, y la descripción larga puede omitirse.

```
<?php
```

```

/*
 * Descripcion corta.
 *
 * Esto forma parte
 *
 * de la descripcion larga.
 *
 */
function x($y)
{
    return abs($y);
}

?>

```

The screenshot shows the phpDocumentor web interface. At the top, it says "Powered by **phpDocumentor** docBuilder :: phpDocumentor v1.4.3 Web Interface". Below is a navigation menu with links: Introduction, Config, Files, Output, Options, Credits, and Links.

Target: C:\wamp\www\PhpDocumentor\docbuilder
Target is the directory where the output produced by phpDocumentor will reside.

Output Format: HTML:Smarty:default
Output format may be HTML, XML, PDF, or CHM (case-sensitive) in version 1.2. There is only one Converter for both CHM and PDF: default.

Output type: Converter name:template name
There are 2 HTML Converters: frames or Smarty.
Smarty templates may be any of: default, earthli, Kib2t, phptocde, phptoclib, phredit, DOM/default, DOM/earthli, DOM/kib2t, or DOM/phptocde.

HTML:Smarty:default
Add the converter in the help box.

Working Directory: C:\wamp\www\PhpDocumentor\docbuilder

A preview window shows a generated HTML page titled "phpDocumentor Guide to Creating Good Documentation". The page contains sections like "Why write good documentation?", "Writing Great Documentation", and "Consider This". It includes sample code and explanations of various documentation styles.

Figura 4. Con PHPDocumentor es posible generar documentación en diferentes formatos de manera sencilla.

Podemos dar un formato específico a las descripciones añadiendo etiquetas o tags al estilo **HTML** en los comentarios, las cuales serán luego interpretadas, según corresponda, por PHPDocumentor:

**** (negrita)
<code> (resaltar código PHP)
**
** (salto de línea)
<i> (italicas)
<kbd> (tecla)
**** (lista)
**** (lista ordenada)
<p> (párrafo)
<pre> (preserva el formato original)
<samp> (resaltar código no PHP)
**** (lista no ordenada)
<var> (nombre de variable)

Algunos de los elementos del lenguaje que podemos incluir en la documentación generada (anteponiendo los comentarios) son:

- Clases (**class**)
- Métodos
- Propiedades
- Constantes (**define**)
- Variables globales
- Inclusiones (**include**, **include_once**, **require**, **require_once**)

```

<?php

/**
 * Archivo que contiene el encabezado del sitio.
 * En formato HTML.
 */
include "templates/encabezado.html";
  
```

III JAVASCRIPT

Para que podamos utilizar PHPDocumentor a través de un navegador, debemos contar con el soporte para JavaScript habilitado, debido a las características de la interfaz para acceder a la aplicación. Hoy en día, los principales navegadores nos dan la posibilidad de desactivar JS y hasta podría ocurrir el caso de que no brinden soporte.

```
/*
 * Constante que contiene el valor de la letra PI
 */
define("PI", 3.14159);

/**
 * La clase calcularSuperficie() devuelve la superficie de una
circunferencia.

*/
class calcularSuperficie {

    /**
     * La propiedad $radio debe ser definida antes de invocar al metodo
     mostrarSuperficie().
    */
    public $radio;

    /**
     * El metodo mostrarSuperficie() imprime la superficie de una
circunferencia.
    *
     * Antes de mostrar la superficie hay que definir la propiedad $radio.
    */
    final function mostrarSuperficie() {
        if (is_numeric($this->radio))
            return pow(PI * $this->radio, 2);
        else
            echo 'Asigne radio';
    }
}
```

III LINEA DE COMANDOS

Podemos generar documentación con PHPDocumentor a través de una interfaz web o desde la línea de comandos que, aunque menos intuitiva, es la única alternativa que tenemos para acceder a determinadas opciones avanzadas de la aplicación, las cuales son listadas en detalle en el sitio oficial de la herramienta.

```

$temp = new calcularSuperficie();

$temp->radio = 6;

echo $temp->mostrarSuperficie();

/**
 * Archivo que contiene el pie de pagina del sitio.
 * En formato HTML.
 */
include "templates/pie.html";

?>

```

The screenshot shows the PHPDocumentor interface with the following details:

- Packages:** default
- Files:** calcularSuperficie.php
- Classes:** calcularSuperficie

Class: calcularSuperficie

Source Location: /calcularSuperficie.php

Class Overview [line 17]

La clase calcularSuperficie() devuelve la superficie de una circunferencia.

Author(s):
Version:
Copyright:

Variables \$radio	Methods mostrarSuperficie
Constants	

Inherited Variables **Inherited Methods**

Inherited Constants

Class Details

La clase calcularSuperficie() devuelve la superficie de una circunferencia.

[Top]

Class Variables

\$radio

[line 22]

La propiedad \$radio debe ser definida antes de invocar al metodo mostrarSuperficie.

Tags:

Figura 5. Cada vez más aplicaciones utilizan PHPDocumentor para generar documentación tanto descargable como de visualización online.

Otros elementos opcionales son las etiquetas, que permiten incluir información adicional. La sintaxis requerida por PHPDocumentor es:

```
* @nombreEtiqueta valor
```

Por ejemplo:

```
/**  
 * Descripcion1.  
 *  
 * Descripcion2.  
 *  
 * @nombreEtiqueta1 valor  
 * @nombreEtiqueta2 valor  
 * @nombreEtiquetaN valor  
 */
```

Entre las etiquetas disponibles encontramos:

@access (tipo de acceso, público o privado)
@author (autor, de la forma **nombreAutor <emailAutor>**)
@copyright (título y fecha de creación)
@deprecated (texto)
@example (ruta a un ejemplo relativo)
@global (variables globales, de la forma **tipoVariable nombreVariable o tipoVariable descripción**)
@internal (información privada)
@param (parámetros, de la forma **tipoParametro [\$nombre] descripción**)
@return (de la forma **tipoDatoDevuelto descripción**)
@link (URL)
@name (nombre)
@package (nombre del paquete)
@see (nombre de otro elemento relativo)

III HOSTING

Debemos saber que es común que en los servicios que ofrecen alojamiento web no se ofrezca la posibilidad de acceder y modificar el archivo llamado **php.ini**, por lo que algunas funciones disponibles en PHP, como **error_reporting** e **ini_set**, nos son de gran ayuda para configurar el nivel de reporte de errores, entre otras opciones que necesitemos.

- @since** (versión o fecha)
- @static** (elementos estáticos)
- @staticvar** (tipo y descripción de una variable estática)
- @todo** (texto)
- @var** (tipo de dato de una propiedad)
- @version** (versión)

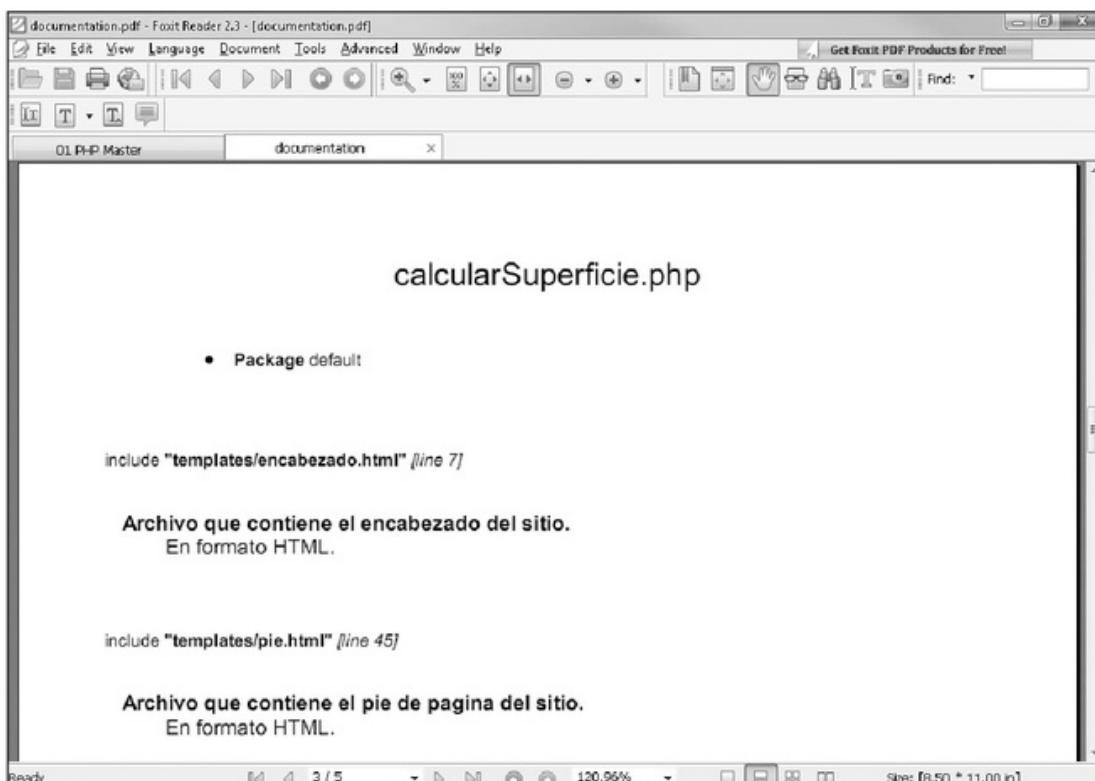


Figura 6. Es habitual que necesitemos generar información de nuestras aplicaciones utilizando el formato PDF debido a sus características y a su gran popularidad.

La aplicación nos proporciona dos formas de uso: desde la línea de comandos o accediendo a través de un navegador web como veremos a continuación. Podremos observar varias opciones para configurar la documentación por generar, entre las que se encuentran:

III NUEVO SISTEMA

Es importante mencionar que en las últimas versiones, el lenguaje de programación PHP admite un nuevo sistema llamado **PhD** (*[PH]P based [D]ocBook renderer*) para generar documentación. Está escrito en PHP, y una de las características más visibles es el resultado de la sintaxis utilizada para realizar la implementación de clases y objetos.

- **Files** (aquí se incluyen los archivos y/o directorios que queremos documentar, y/o los archivos por omitir).
- **Output** (se definen el directorio y el formato de salida).
- **Options** (opciones variadas como título de la documentación, etiquetas adicionales, resaltado de código, etcétera).

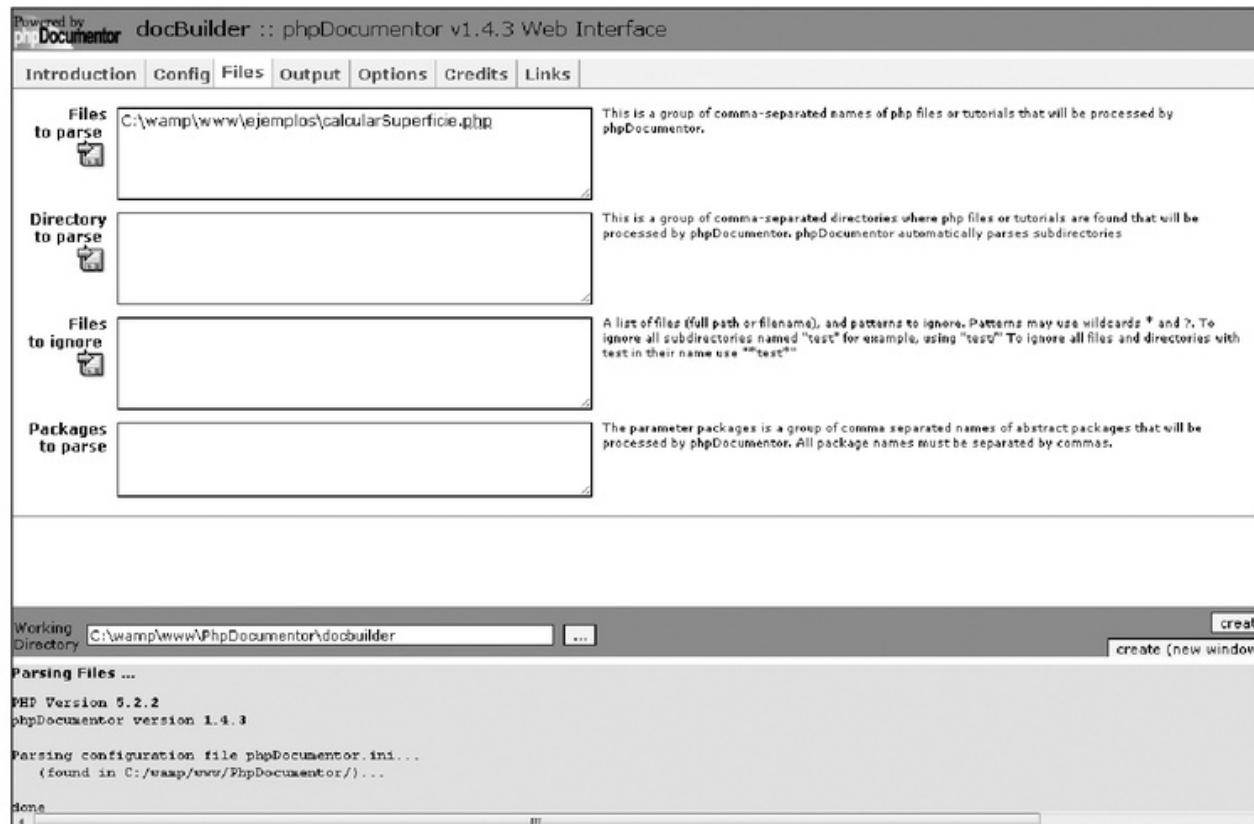


Figura 7. La interfaz que nos ofrece PHPDocumentor es intuitiva y admite múltiples opciones para personalizar los resultados.

Una vez definidas estas opciones, debemos presionar el botón **create**. Los resultados de la operación se verán en la parte inferior de nuestro monitor. Si todo funcionó correctamente, la documentación generada se mostrará almacenada (en el/los formato/s seleccionado/s) en el directorio indicado.

III CÓDIGO LIMPIO

Debemos saber que hay disponibles muchas clases para clarificar el código fuente de una aplicación, y PEAR ofrece una llamada **PHP_CodeSniffer**, la cual nos permite, entre otras cosas, identar líneas y prevenir la utilización indebida de palabras reservadas. Funciona desde la línea de comandos y se descarga desde <http://pear.php.net>.

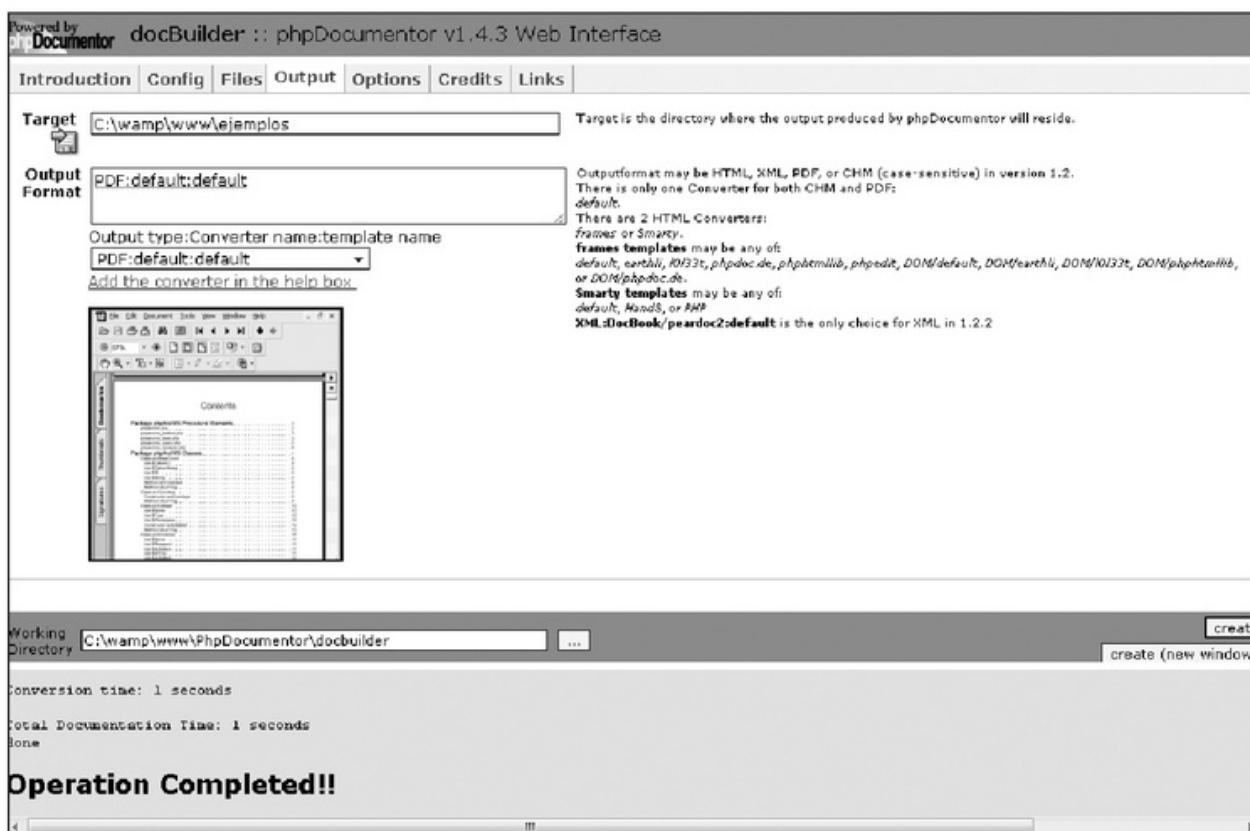


Figura 8. Podemos obtener más información acerca de PHPDocumentor en su sitio oficial, en donde se incluye información de última hora y ejemplos de aplicación.

MANEJO Y CONTROL DE ERRORES

Mantener el control sobre un sistema en etapa de desarrollo implica que conozcamos de manera clara lo que ocurre, para capturar y solucionar los inconvenientes. PHP incluye ciertas funcionalidades que permiten visualizar los errores que se suceden y que nos ayudan, a través de mensajes claros, a detectar cuáles son las fallas y en qué parte del código se producen.

Se estila mantener activadas las opciones para informe de errores en la etapa de desarrollo de sitios y no en la de producción. Esto es por dos razones principales: no afectar el estilo de nuestras páginas y no brindar información que podría ser útil a terceros para fines diversos.

Directivas de configuración

En el archivo principal de configuración de PHP, **php.ini**, se incluyen directivas para definir distintas opciones en lo referido a la visualización y registro de errores, entre las que podemos encontrar:

```

error_reporting
display_errors
display_startup_errors
log_errors
error_log
log_errors_max_len
ignore_repeated_errors
ignore_repeated_source
track_errors
html_errors
error_prepend_string
error_append_string

```

PHP maneja diversos tipos o niveles de error referenciados a través de constantes, y, con **error_reporting**, es posible definirlos mediante el uso de los operadores disponibles.

CONSTANTE	DESCRIPCIÓN
E_ALL	Nos muestra todos los errores y advertencias, excepto los correspondientes a E_STRICT .
E_COMPILE_ERROR	Emite mensajes acerca de errores fatales en tiempo de compilación.
E_COMPILE_WARNING	Es similar al anterior, sólo que se encarga de mostrar únicamente las advertencias ante eventuales errores.
E_CORE_ERROR	Trata errores fatales producidos durante el inicio de PHP (carga de librerías, directivas, etcétera).
E_CORE_WARNING	Es similar al anterior, sólo que se encarga de mostrar únicamente las advertencias ante eventuales errores.
E_ERROR	Muestra mensajes acerca de errores fatales en tiempo de ejecución. Si omitimos esta opción, los errores fatales serán de igual manera exhibidos.
E_NOTICE	Se encarga de los errores no críticos en el código fuente de la aplicación (variables no inicializadas, índices de arrays inexistentes, etcétera).
E_PARSE	Emite advertencias ante errores de compilación internos.
E_STRICT	Está disponible a partir de PHP 5 y se enfoca en la compatibilidad del código escrito con relación a versiones anteriores del lenguaje.
E_USER_ERROR	Es similar a E_ERROR , sólo que podemos personalizar los mensajes.
E_USER_NOTICE	Es similar a E_NOTICE , sólo que podemos personalizar los mensajes.
E_USER_WARNING	Es similar a E_WARNING , sólo que podemos personalizar los mensajes.
E_WARNING	Imprime advertencias en tiempo de ejecución y errores no fatales.

Tabla 1. Opciones para la directiva *error_reporting*.

Cada nivel tiene su propósito. El de **E_NOTICE**, por ejemplo, es el de enumerar ciertas situaciones en las que nuestro código funciona, pero tiene inconsistencias que

pudieron haber sido notadas o no. Es muy probable que cualquier sistema, ya sea en etapa de desarrollo o de producción, sufra las notificaciones de **E_NOTICE**. A partir de PHP versión 6, se incluirá el nivel de errores denominado **E_RECOVERABLE_ERROR**, el cual podremos utilizar en lugar de **E_ERROR** para identificar errores que requieren un control y un manejo específicos, pero de los que el sistema puede recuperarse, o sea, que produzcan una inestabilidad no definitiva. Además, el nivel de errores **E_ALL** contendrá al nivel **E_STRICT**.

Contamos con los siguientes operadores para combinar las opciones disponibles:

- ~ (negación)
- | (alternativa)
- & (concatenación)

```
error_reporting = E_ALL
```

```
error_reporting = E_ALL & ~E_NOTICE
```

```
error_reporting = E_ALL & ~(E_NOTICE | E_WARNING)
```

La directiva **display_errors** nos permite definir si se emitirán mensajes de error a través de la salida estándar (probablemente, un navegador web) o no. Su valor de manera predeterminada es 1. En el mismo sentido, **display_startup_errors** nos muestra errores producidos durante la carga de PHP (por ejemplo, extensiones no encontradas, dependencias erróneas, directorios no hallados, etcétera), que no están contenidos entre los que se muestran configurando la directiva **display_errors**. Podemos asignarle los valores 1 (verdadero) o 0 (falso, predeterminado). Los registros generados por el servidor web nos pueden servir de gran ayuda a la hora de capturar y de reconocer errores. Para registrarlos, existe la directiva **log_errors**.

III EXCEPCIONES

PEAR (<http://pear.php.net>), el repositorio oficial de extensiones y aplicaciones para PHP, ofrece una clase específica para las versiones 5 y superiores llamada **PEAR_Exception**. Esta clase nos permite realizar la implementación de excepciones. Veremos más información acerca de las características de PEAR en el **Capítulo 8**, más adelante en este libro.

que, de manera predeterminada, está deshabilitada. Podemos visualizar los errores con el detalle de la fecha, la hora, nivel de error, y descripción. A su vez, también registra errores de sintaxis. En la instalación clásica de **Apache**, podremos ubicar este archivo dentro del directorio **logs**. El documento que contiene los errores, normalmente, se llama **error.log** y, al ser un archivo de texto plano, podemos visualizarlo con cualquier editor (para esto el servidor deberá estar inactivo).

```

apache_error.log - Bloc de notas
Archivo Edición Fichero Ayuda
[Thu May 24 23:12:20 2007] [notice] Apache/2.2.4 (Win32) PHP/5.2.2 configured -- resuming normal operations
[Thu May 24 23:12:20 2007] [notice] Server built: Jan 9 2007 23:12:20
[Thu May 24 23:12:20 2007] [notice] Parent: Created child process 2929
[Thu May 24 23:12:20 2007] [notice] Child 2929: Child process is running
[Thu May 24 23:12:20 2007] [notice] Child 2929: Acquired the start mutex
[Thu May 24 23:12:20 2007] [notice] Child 2929: Starting 250 worker threads.
[Thu May 24 23:12:20 2007] [notice] Child 2929: Waiting for thread to listen on port 80.
[Thu May 24 23:12:24 2007] [error] [client 127.0.0.1] File does not exist: C:/wamp/www/favicon.ico
[Thu May 24 23:12:24 2007] [error] [client 127.0.0.1] File does not exist: C:/wamp/www/favicon.ico
[Thu May 24 23:12:24 2007] [error] [client 127.0.0.1] File does not exist: C:/wamp/www/favicon.ico
[Thu May 24 23:12:24 2007] [notice] Parent: Received shutdown signal -- shutting down the server.
[Thu May 24 23:12:24 2007] [notice] Child 2929: Exit event signaled, child process is ending.
[Thu May 24 23:12:24 2007] [notice] Child 2929: Released the start mutex
[Thu May 24 23:21:09 2007] [notice] Child 2929: Waiting for 250 worker threads to exit.
[Thu May 24 23:21:10 2007] [notice] Child 2929: All worker threads have exited.
[Thu May 24 23:21:10 2007] [notice] Child 2929: Child process is exiting.
[Thu May 24 23:21:11 2007] [notice] Parent: Child process exited successfully.
[Thu May 24 23:21:11 2007] [notice] Apache/2.2.4 (Win32) PHP/5.2.2 configured -- resuming normal operations
[Thu May 24 23:21:11 2007] [notice] Server built: Jan 9 2007 23:12:20
[Thu May 24 23:21:18 2007] [notice] Parent: Created child process 1752
[Thu May 24 23:21:18 2007] [notice] Child 1752: Child process is running
[Thu May 24 23:21:18 2007] [notice] Child 1752: Acquired the start mutex
[Thu May 24 23:21:18 2007] [notice] Child 1752: Starting 250 worker threads.
[Thu May 24 23:21:18 2007] [notice] Child 1752: Waiting for thread to listen on port 80.
[Thu May 24 23:21:40 2007] [notice] Parent: Received shutdown signal -- shutting down the server.
[Thu May 24 23:21:40 2007] [notice] Child 1752: Exit event signaled, child process is ending.
[Thu May 24 23:21:42 2007] [notice] Child 1752: Released the start mutex
[Thu May 24 23:21:42 2007] [notice] Child 1752: Waiting for 250 worker threads to exit.
[Thu May 24 23:21:42 2007] [notice] Child 1752: All worker threads have exited.
[Thu May 24 23:21:42 2007] [notice] Child 1752: Child process is exiting.
[Thu May 24 23:21:42 2007] [notice] Parent: Child process exited successfully.
[Thu May 25 13:32:17 2007] [notice] Apache/2.2.4 (Win32) PHP/5.2.2 configured -- resuming normal operations
[Thu May 25 13:32:17 2007] [notice] Server built: Jan 9 2007 23:12:20
[Thu May 25 13:32:17 2007] [notice] Parent: Created child process 2440
[Thu May 25 13:32:17 2007] [notice] Child 2440: Child process is running
[Thu May 25 13:32:17 2007] [notice] Child 2440: Acquired the start mutex
[Thu May 25 13:32:17 2007] [notice] Child 2440: Starting 250 worker threads.
[Thu May 25 13:32:17 2007] [notice] Child 2440: Starting thread to listen on port 80.
[Thu May 25 13:34:26 2007] [error] [client 127.0.0.1] File does not exist: C:/wamp/www/bs/desarrollo/admin/css/style2.css
[Thu May 25 13:34:26 2007] [error] [client 127.0.0.1] File does not exist: C:/wamp/www/bs/desarrollo/admin/products.php
[Thu May 25 13:34:26 2007] [notice] Parent: Received shutdown signal -- Shutting down the server.
[Thu May 25 13:34:26 2007] [notice] Child 2440: Exit event signaled, child process is ending.
[Thu May 25 13:34:26 2007] [notice] Child 2440: Released the start mutex
[Thu May 25 13:34:26 2007] [notice] Child 2440: Waiting for 250 worker threads to exit.
[Thu May 25 13:34:26 2007] [notice] Child 2440: All worker threads have exited.
[Thu May 25 13:34:27 2007] [notice] Child 2440: Child process is exiting.
[Thu May 25 13:34:27 2007] [notice] Parent: Child process exited successfully.
[Thu May 25 12:24:47 2007] [notice] Apache/2.2.4 (Win32) PHP/5.2.2 configured -- resuming normal operations
[Thu May 25 12:24:47 2007] [notice] Server built: Jan 9 2007 23:12:20

```

Figura 9. Los archivos de configuración nos permiten mantener un control sobre los sucesos que acontecen en la vida diaria de un sistema.

Para almacenar los errores en un archivo diferente de **error.log** (por caso para no utilizar el mismo archivo de errores del servidor web), tendremos que modificar la directiva **error_log**, especificando la ruta hacia él.

Si necesitamos determinar o limitar el tamaño máximo en bytes del registro, tenemos que definir la directiva **log_errors_max_len**. Su valor de manera predeterminada es de 1024 bytes. Si no queremos imponer ningún límite, simplemente, le asignamos 0.

Con **ignore_repeated_errors** habilitada, PHP nos mostrará los mismos errores sólo una vez. Para considerarse repetidos, deben ocurrir en la misma línea y en el mismo archivo (normalmente estarán dentro de ciclos **for**, **while**, **do while**, **foreach**, etcétera). En relación a esto, la directiva **ignore_repeated_source** modifica la definición de lo que es un error repetido: **ignore_repeated_source** activada no tiene en cuenta el origen de los errores, es decir, el archivo en el que se produjeron.

El lenguaje PHP incorpora la variable **php_errormsg**, que contiene de manera predeterminada el mensaje correspondiente al último error cometido. Para habilitar

Este comportamiento, debemos activar la directiva **track_errors**, que está deshabilitada de manera predeterminada:

```
<?php

$texto = @file_get_contents($archivo);

if ($php_errormsg) {
    echo 'Ocurrio un error.';
    die();
}

echo 'Todo OK.';

?>
```

Entre las directivas para dar formato a los mensajes de error generados, está disponible **html_errors**. Si esta directiva se encuentra habilitada, los errores se muestran como mensajes en HTML con enlace a la descripción contenida en el manual de PHP (la ruta al manual se define en **docref_root** y la extensión, desde **docref_ext**). Para anexar contenidos antes y después de los mensajes de error, existen las directivas **error_prepend_string** y **error_append_string** respectivamente, que nos permitirán personalizar el formato o la presentación de los mensajes de error y adecuarlos a nuestras necesidades.

Funciones

El lenguaje PHP incorpora diversas funciones para que podamos administrar el muestreo de errores entre las que se incluyen **ini_set** y **error_reporting**. Los argumentos a **ini_set** son el nombre de la directiva del **php.ini** que queramos modificar y el valor correspondiente:

III SEGURIDAD

Debemos destacar que la seguridad en las aplicaciones web es un aspecto directamente ligado a la captura y solución de los errores que puedan presentarse. En el **Capítulo 7** de éste libro, veremos cómo los mensajes de error pueden ser de utilidad para modificar el correcto comportamiento de los sistemas y averiguar su arquitectura interna.

```
ini_set('display_errors', '1');
```

El cambio sólo tendrá validez mientras dure el script y, por cuestiones de seguridad, no todas las directivas pueden modificarse. Podremos encontrar más información acerca del detalle de las directivas permitidas en el manual de PHP (www.php.net).

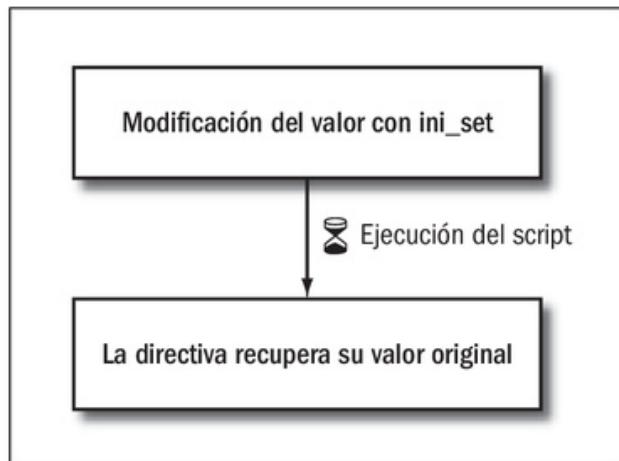


Figura 10. Las modificaciones realizadas mediante la función *ini_set* sólo tendrán vigencia durante la ejecución del script.

En cuanto a **error_reporting**, recibe como argumento el nivel de error que queremos establecer. Los valores aceptados son:

VALOR	CONSTANTE DE LA DIRECTIVA
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE
2047	E_ALL
2048	E_STRICT

Tabla 2. Opciones para la función *error_reporting*.

Para seleccionar varias opciones al mismo tiempo, como en la directiva del mismo nombre, en lugar de utilizar operadores debemos sumar los valores enteros corres-

pondientes a las constantes. Por ejemplo, si quisiéramos utilizar las opciones **E_ERROR**, **E_WARNING** y **E_PARSE**:

```
error_reporting(7);
```

Lo que sería equivalente a la siguiente línea:

```
ini_set('error_reporting', 'E_ERROR & E_WARNING & E_PARSE');
```

Esta función devuelve el valor anterior de la directiva.

Otra función que podríamos sumar es **die**, que emite un mensaje personalizado y finaliza el script actual:

```
<?php

$id = $_GET['id'] != '' ? $_GET['id'] : die('id no recibido !');
echo "id: $id";
?>
```

Es de propósito general, no orientada exclusivamente al tratamiento de errores, pero también puede aplicarse.

Bases de datos

El acceso a la información contenida en una base es algo común hoy día para cualquier clase de aplicaciones, ya sea en simples portales con posibilidades de registro hasta complejos sistemas de negocios, por lo que cada extensión mantiene su propio sistema gestor de errores, ya sea a través de funciones como de métodos de objetos.

III PERROR

Debemos saber que una aplicación de MySQL que nos servirá para interpretar mejor los mensajes de error es la denominada **perror**. Gracias a ella podemos obtener más información o descripciones acerca de los errores, podemos utilizar este programa, que viene junto con la distribución de MySQL, y se invoca desde la línea de comandos.

MySQL provee la función **mysql_error**, que nos devuelve una descripción del último error cometido. Recibe como argumento opcional un identificador de conexión (si se opta por no incluirlo, la función intenta encontrar la última conexión abierta activa al servidor y, si no la halla, intenta crear una nueva):

```
echo mysql_error();
```

En MySQL los errores están asociados a un código y es posible conocer este código utilizando la función **mysql_errno**, que también recibe como argumento opcional un identificador de conexión:

```
echo mysql_errno();
```

La directiva del archivo **php.ini mysql.trace_mode** nos permite visualizar por pantalla advertencias y errores relacionados a SQL.

Siempre existe la posibilidad de que capturemos errores a través de excepciones, como veremos en la próxima sección:

```
try {  
  
    if (!@mysql_connect('localhost', 'user', 'pass'))  
        throw new Exception (mysql_error());  
    } catch (Exception $e) {  
        echo 'Error en la conexión. ';  
    }
```

La extensión **mysqli** mantiene sus propias funciones y avanza sobre funcionalidades no ofrecidas por la extensión **mysql** tradicional, como por ejemplo:



III MYSQL Y PHP, UNA LARGA HISTORIA JUNTOS

Es importante recordar que MySQL es considerado por muchos el mejor motor de bases de datos para proyectos web. Se ha ganado este puesto por sus livianos requerimientos (sobre todo en sus primeras versiones), MySQL es descrito y ofrecido por usuarios y empresas como motor preferencial para los sitios desarrollados en PHP.

- errores en el establecimiento de una conexión;
- en consultas preparadas;
- el último error general ocurrido.

Para el primer caso, disponemos de las funciones **mysqli_connect_error** (nos permite obtener la descripción del último error cometido o bien una cadena vacía si no se cometió ningún error) y **mysqli_connect_errno** (nos permite conocer el código del último error cometido o bien cero si no hubo ningún error).

Respecto de las consultas preparadas, contamos con las funciones **mysqli_stmt_errno** (código de error) y **mysqli_stmt_error** (descripción). Por su parte, la función **mysqli_stmt_sqlstate** nos devuelve un código de error (o 00000 si no han ocurrido errores) correspondiente a la última operación realizada.

De forma general, con la función **mysqli_warning_count** obtenemos el número de advertencias generadas a partir de la última consulta y con **mysqli_sqlstate**, un código de error (o 00000 si no han ocurrido) de la última operación realizada.

Por su parte, la cada vez más popular extensión **sqlite** para acceder a bases de datos **SQLite** nos ofrece sus propias funciones/métodos para reconocer errores. La función **sqlite_open** admite como tercer argumento imprimir un mensaje relacionado con el error producido en caso de no poder abrir (o crear) la base de datos.

```
if (!$db = sqlite_open("db.sqlite", 0666, $error)) {
    echo $error;
    exit;
}
```

La función **sqlite_busy_timeout** establece el tiempo máximo en milisegundos para conectarse a una base de datos luego del cual devolverá error.

Recibe como argumentos un manejador de base de datos y el número de milisegundos (por defecto este valor es 60000, o sea 60 segundos).

Podemos obtener el código de error producido mediante **sqlite_last_error** (que recibe como argumento un manejador de base de datos) y la descripción del error a través de la función **sqlite_error_string** (que recibe como argumento el código).

Tanto la extensión **mysqli** como **sqlite** mantienen la alternativa orientada a objetos, por lo que todas las funciones aquí descriptas tienen sus métodos correspondientes disponibles.

Excepciones

En el marco de un sistema, una excepción es un suceso que interrumpe la ejecución normal de un programa. El soporte para manejo de excepciones en PHP viene evolucionando desde hace varias versiones y desde PHP 5 contamos con los bloques **try/catch**, para intentar capturar y lanzar excepciones para el control de errores. Las funciones disponibles para tratamiento y control de excepciones son:

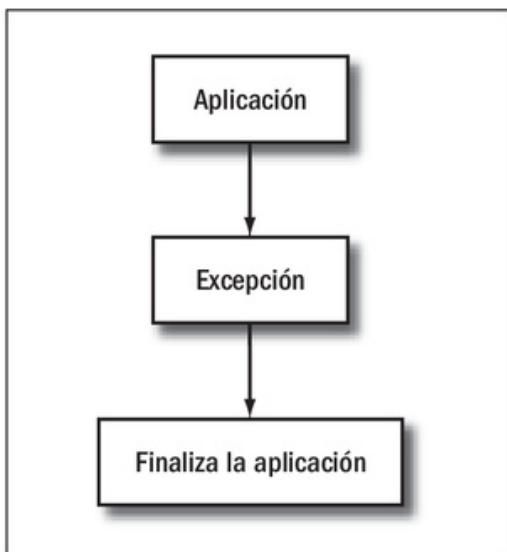


Figura 11. Una excepción no capturada hace que la aplicación finalice de inmediato.

trigger_error: genera un mensaje de error de usuario. Sólo algunos niveles de error pueden ser interceptados por esta función (**E_USER_NOTICE**, **E_USER_WARNING**, y **E_USER_ERROR**):

```

<?php

if (!$flag) {
    trigger_error("Aqui el mensaje personalizado", E_USER_NOTICE);

}
?>
  
```

set_error_handler: nos permite cambiar el manejador de errores predeterminado de PHP por uno personalizado.

set_exception_handler: crea una función de respuesta para aquellas situaciones en las que una excepción se produzca sin haber sido capturada:

```
<?php
```

```

set_error_handler("abrirArchivo");

try {
    fopen('nombreArchivoInexistente.conf', 'r');
} catch (Exception $error) {
    echo $error->getMessage();
}

restore_error_handler();

function abrirArchivo($codigoError, $mensajeError) {
    throw new Exception("Imposible abrir el archivo.");
}

?>

```

Mediante la función **set_error_handler**, establecemos una manera de gestionar los errores producidos en el bloque, transfiriendo el control a la función **abrirArchivo**. Con **restore_error_handler**, devolvemos el control a PHP.

```

<?php

function buscarCC($i) {
    set_error_handler("manejadorBuscarCC");

    $cc[0] = 'Visa';
    $cc[1] = 'Mastercard';

    try {
        echo "<li>Tarjeta seleccionada: ".$cc[$i]."<br>";
    } catch (Exception $e) {
        echo "<li><b>Excepcion: ".$e->getMessage()."</b>";
    }
}

function manejadorBuscarCC($errno, $errmsg) {
    throw new Exception("Opcion inexistente<br>");
}

```

```

buscarCC(0);
buscarCC(1);
buscarCC(2);

?>

```

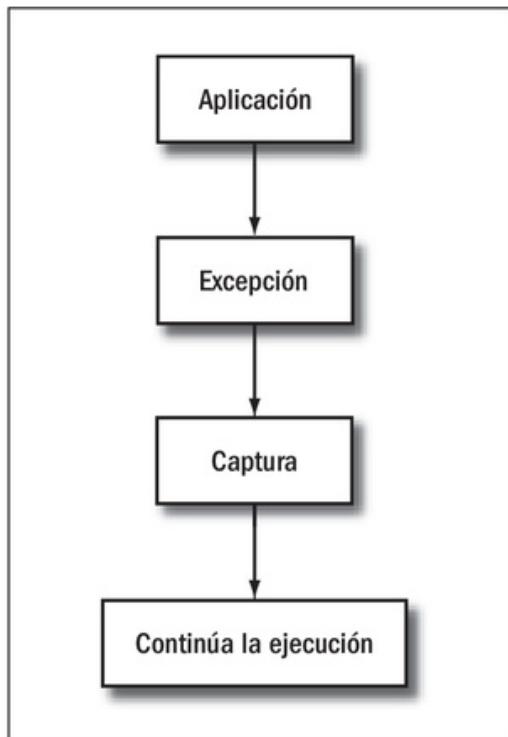


Figura 12. Las funciones puestas a disposición del desarrollador por parte de PHP permiten capturar excepciones manteniendo la integridad de las aplicaciones.

Disponemos de otros métodos para el tratamiento de excepciones:

getCode: nos indica el código de error.

getFile: nos muestra la ruta al archivo en el cual se generó la excepción.

getLine: nos devuelve la línea en la cual se generó la excepción.

getTrace: recupera un array multidimensional con información acerca del desarrollo



MÁS SOBRE EXCEPCIONES ONLINE

Debemos saber que el manual en línea de PHP se encarga de ofrecernos en diversos idiomas una detallada referencia sobre el uso de excepciones. Podemos encontrar el sitio en: www.php.net/exceptions. A su vez, hallaremos un breve resumen sobre excepciones en: www.w3schools.com/php/php_exception.asp.

de la excepción. El método **getTraceAsString** nos devuelve la misma información en formato de cadena de caracteres.

Más allá de los ejemplos, normalmente, utilizaremos las excepciones para capturar y tratar errores externos, o sea, los que no tienen que ver en principio con la lógica de nuestros programas (como una conexión a una base de datos que no responde o un servicio web que no puede resolver una petición). Encontramos más información en la siguiente sección del manual oficial de PHP: www.php.net/errorfunc.



RESUMEN

El desarrollo de sistemas es una tarea que se perfecciona a través de la experiencia y de los conocimientos adquiridos. En este capítulo, intentamos exponer ciertas recomendaciones y comentarios acerca de cómo encarar proyectos que sean de fácil manutención y escalables a lo largo del tiempo. El control de errores, la documentación de aplicaciones y la sintaxis utilizada en la escritura de código fueron algunos de los temas que tratamos.



ACTIVIDADES

PREGUNTAS TEÓRICAS

- 1** ¿Por qué es importante documentar un sistema?
- 2** ¿Por qué es importante mantener el código comprensible?
- 3** ¿Un código simple es necesariamente un código vulnerable?
- 4** Busque información acerca de las herramientas para debug en PHP.
- 5** Lea el archivo de errores del servidor. ¿Cuál es el error más común?
- 6** Revise el archivo de errores de PHP. ¿Cuál es el error más común?
- 7** ¿Qué es una excepción?
- 8** Nombre casos en los que resulta necesario el control de excepciones.
- 9** ¿Para qué sirve la directiva `error_reporting`? ¿Y la función?
- 10** ¿Para qué sirve la función `ini_set`?

EJERCICIOS PRÁCTICOS

- 1** Desarrolle una aplicación y genere la documentación correspondiente.
- 2** Tome un sistema ya desarrollado y modifíquelo para luego generar la documentación correspondiente.
- 3** ¿Qué nivel de error utilizaría en la etapa de desarrollo? ¿Por qué?
- 4** ¿Qué nivel de error utilizaría en la etapa de producción? ¿Por qué?
- 5** ¿Qué incorporan las versiones de PHP 5 y superiores en relación al manejo de errores?

Templates

En este capítulo, veremos dos motores de templates altamente aceptados por la comunidad de desarrolladores y con diferentes grados de complejidad: Sigma y Smarty. Una vez comprendido su funcionamiento, podremos desarrollar aplicaciones con mucha más agilidad.

Introducción	40
Sigma	40
Templates preparados	43
Variables	43
Bloques	45
Inclusión de archivos externos	50
Funciones	52
Otros métodos disponibles	53
Ejemplo completo de uso	53
Smarty	61
Modos de trabajo	61
Variables	63
Comentarios	66
Archivos de configuración	66
Propiedades y métodos del objeto Smarty	70
Operadores	71
Condicionales	72
Ciclos	72
Inclusión de archivos externos	74
Funciones	74
Caché	82
Objetos	83
Ejemplo completo de uso	84
Resumen	91
Actividades	92

INTRODUCCIÓN

Separar la lógica del diseño en una aplicación es una regla casi obligada en desarrollos profesionales. Esta actividad facilita el proceso de creación de sitios haciéndolos modulares y fácilmente mantenibles. Así, evita trabajo extra tanto a los desarrolladores como a los diseñadores de sitios web.

Contamos con una gran cantidad de opciones en el mercado (incluso son muchos los que crean sus propias implementaciones sólo para uso personal y las adecuan a sus necesidades). Para este capítulo, elegimos **Sigma**, una de las alternativas ofrecidas por PEAR, y **Smarty**, desarrollada y mantenida por algunos de los responsables de PHP. Mientras Sigma nos ofrece un soporte sencillo y práctico para templates, Smarty va más allá y brinda multitud de opciones tanto para programadores como para diseñadores, alejándose quizás del propósito inicial de una herramienta de este tipo.

SIGMA

PEAR (<http://pear.php.net>) nos ofrece varias alternativas en lo referido a la implementación de templates. Sigma (el paquete se denomina **HTML_Template_Sigma**) es una de ellas. Deriva de otra llamada **IT** (**HTML_Template_It**), al tiempo que hereda todas sus características y suma funcionalidades.

The screenshot shows the PEAR package page for **HTML_Template_Sigma**. At the top, there's a navigation bar with links for Main, Support, Documentation, Packages, Package Proposals, Developers, and Bugs. Below the navigation is a search bar. The main content area has a title "Package Information: HTML_Template_Sigma". It features several tabs: Main (which is selected), Download, Documentation, Bugs, and Trackbacks. Under the Main tab, there are sections for "Summary", "Current Release", "Description", "Features", "Maintainers", and "More Information". The "Summary" section includes a brief description and links to "PHP License" and "Bug Summary". The "Current Release" section lists "1.2.0 (stable)" released on "2006-07-22" with a link to the "Changelog". The "Description" section notes that it implements the Integrated Templates API. The "Features" section lists various capabilities like nested blocks, automatic removal of empty blocks, and support for regular expressions. The "Maintainers" section lists "alevey Borzov (lead)". The "More Information" section provides links to "Browse the source tree", "RSS release feed", and "View PEAR-devel statistics".

Figura 1. Sigma es uno de los paquetes puestos a disposición del público en el sitio de PEAR. Allí podremos encontrar un manual detallado con todos sus métodos y propiedades.

Esta herramienta nos brinda una clase base llamada **HTML_Template_Sigma**. Para disponer de sus métodos y propiedades, tendremos, necesariamente, que incluir el archivo **Sigma.php** e instanciar un objeto:

```
<?php

require_once 'HTML/Template/Sigma.php';
$template = new HTML_Template_Sigma();

?>
```

El constructor recibe dos argumentosopcionales. El primero es el directorio donde se encuentran los templates, que por defecto es el actual. Esto puede ser modificado estableciendo el directorio en la función constructora.

```
<?php

require_once 'HTML/Template/Sigma.php';
$template = new HTML_Template_Sigma('./html/templates/');

?>
```

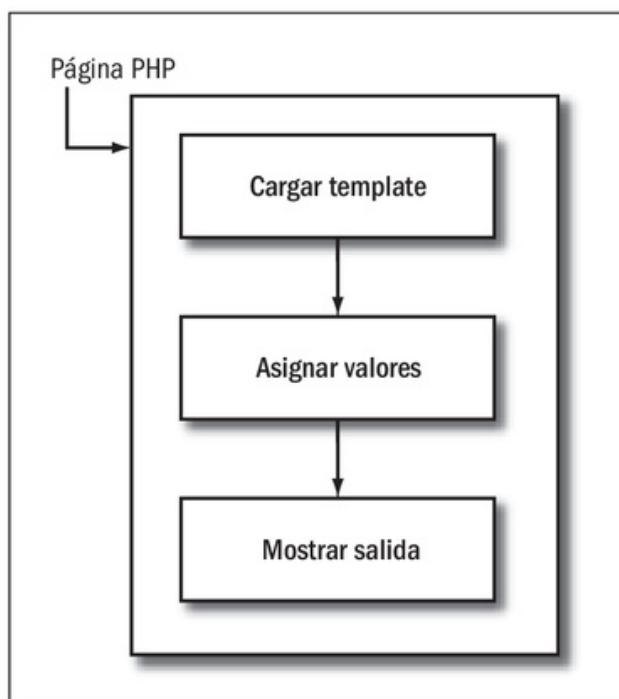


Figura 2. La utilización de templates nos permite estructurar de forma ordenada y clara sitios con múltiples páginas.

En lugar de pasarlo como argumento al constructor, Sigma pone a nuestra disposición el método **setRoot**, que nos permite definir la ruta a los archivos:

```
<?php  
  
require_once 'HTML/Template/Sigma.php';  
$template = new HTML_Template_Sigma();  
$template->setRoot('./html/templates/');  
  
?>
```

El segundo argumento al constructor es el directorio para los templates preparados (opcionalmente podemos utilizar el método **setCacheRoot**).

Para incluir un template desde una página, contamos con el método **loadTemplateFile**, que recibe como argumento la ruta al template. Si no determinamos una ruta absoluta, los templates se buscarán en el directorio definido en el constructor:

```
$template->loadTemplateFile('template.html');
```

Cada template está formado por dos elementos fundamentales: las **variables** y los **bloques**. Un bloque sirve para delimitar un fragmento del template y en su interior puede contener una o más variables, o bloques. Si un bloque sólo contiene variables vacías, se considera vacío. El método **loadTemplateFile** cuenta además con dos argumentos opcionales: el primero indica si se van a omitir las variables sin valores asignados (**true/false**) y el segundo, si se mostrarán o no los vacíos (**true/false**). En ocasiones, necesitaremos incluir un template a partir de una cadena de caracteres, y para esto contamos con el método **setTemplate**:

```
$template->setTemplate($html);
```



EJEMPLOS EN INTERNET

Es importante recordar que podemos obtener el código fuente completo de todos los ejemplos que se incluyen en este libro en el sitio oficial de la editorial: <http://onweb.redusers.com>. Desde allí es posible descargar los proyectos y ponerlos en práctica para tomarlos como base en los futuros desarrollos que decidamos emprender.

Templates preparados

Para no parsear (procesar) el template una y otra vez ante cada requerimiento, contamos con la opción de recurrir a una caché generada internamente por la aplicación. Los datos no son almacenados, sino sólo una representación serializada del template, lo que deriva en una carga más rápida de las páginas en el navegador del usuario, y a una mejor experiencia de uso.

En Sigma, no existe el concepto **TTL** (*Time To Live*, Tiempo De Vida): la actualización de la caché se produce cuando se modifica el template original.

La caché comienza a utilizarse cuando se define un directorio (con permisos para escritura) mediante el constructor o con **setCacheRoot**.

Una vez hecho esto, y ante cualquier llamada a los métodos **loadTemplatefile**, **addBlockfile**, **replaceBlockfile** o a la construcción **INCLUDE** (que veremos más adelante en este capítulo), se buscará una versión disponible del template en el directorio definido automáticamente.

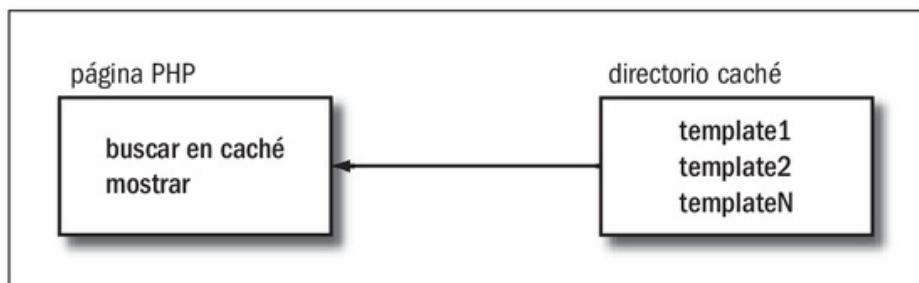


Figura 3. El uso de templates residentes de la caché nos permite reducir el tiempo de carga inicial.

Variables

Una variable se presenta encerrada entre llaves, y su nombre no puede incluir espacios (sí, números, caracteres y guiones):

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE> {titulo} </TITLE>
  </HEAD>
  <BODY>

    {contenido}

  </BODY>
</HTML>

```

Para asignar valores a las variables, contamos con los métodos **setVariable** y **setGlobalVariable** (idéntico al anterior, sólo que genera variables globales). Si el primer argumento a **setVariable** es un nombre de variable, el segundo deberá ser el valor que queremos asignarle. Si en cambio es un array, las claves deberán ser nombres de variables con los valores correspondientes:

```
$variable = 'titulo';
$valor = 'Este es el titulo de la pagina';
$template->setVariable($variable, $valor);

$variable = 'contenido';
$valor = ucwords('la hora actual es: ').date('h:i A');
$template->setVariable($variable, $valor);
```

The screenshot shows the PEAR website interface. At the top, there's a navigation bar with links for Main, Support, Documentation, Packages (which is highlighted), Package Proposals, Developers, and Bugs. Below the navigation bar, there are links for List Packages, Search Packages, Statistics, and Channels. The main content area is titled "Search" and has a sub-section titled "Packages". A search form is present with a text input containing "template", a dropdown menu set to "in Packages", and a "Search" button. Below the search form, the text "Results 1 - 7 of 7:" is displayed, followed by a numbered list of seven packages: 1. HTML_Template_Flexy, 2. HTML_Template_IT, 3. HTML_Template_PHPLIB, 4. HTML_Template_PHPТАL, 5. HTML_Template_Sigma, 6. HTML_Template_Xipe, and 7. URI_Template. At the bottom of the page, there's a copyright notice "Copyright © 2001-2010 The PHP Group All rights reserved." and a link to "Bandwidth and License".

Figura 4. Además de Sigma, PEAR nos ofrece múltiples alternativas en lo referido a la implementación de templates.



TEMPLATES DE PEAR

Debemos recordar que las soluciones brindadas por PEAR son de propósito general y están diseñadas de tal manera que su inclusión en un sitio pueda realizarse de la forma más sencilla posible, manteniendo, de esta forma, una línea clara de trabajo. Los motores de templates son sólo una pequeña parte de lo que PEAR nos puede ofrecer.

```
$variables['titulo'] = 'Este es el titulo de la pagina';
$variables['contenido'] = ucwords('la hora actual es: ').date('h:i A');
$template->setVariable($variables);
```

En caso de que no asignemos valores (una llamada a **setVariable** o a **setGlobalVariable** en cualquier caso suponen una asignación), de manera predeterminada la variable es removida, es decir, reemplazada por un valor nulo.

Durante la ejecución de un script, las variables pueden llegar a tomar distintos valores. Es posible **limpiar** el contenido de ellas mediante el método **clearVariables**. Esto no afecta a las globales, que retendrán su contenido a pesar de esta instrucción. Una vez que asignamos valores a las variables, el método **parse** realiza las sustituciones correspondientes. Luego, a través del método **get**, podemos recuperar la salida generada de la siguiente forma:

```
$template->parse();
$html = $template->get();
```

En lugar de recuperar el contenido parseado directamente, es posible enviar la salida al navegador a través del método **show**:

```
$template->show();
```

Bloques

Un bloque sirve para encerrar o delimitar un fragmento dentro de un template. En su interior podemos ubicar una o más variables o bloques. Su sintaxis es la siguiente:

```
<!-- BEGIN nombreBloque -->
aqui el contenido del bloque
<!-- END nombreBloque -->
```

Por ejemplo, si queremos incluir filas de una tabla HTML dentro del bloque **nombreBloque** lo haríamos de la siguiente manera:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
```

```

<HEAD>
  <TITLE> {titulo} </TITLE>
</HEAD>

<BODY>

<TABLE>
<!-- BEGIN nombreBloque -->
<TR>
  <TD>{columna1}</TD>
  <TD>{columna2}</TD>
</TR>
<!-- END nombreBloque -->
</TABLE>

</BODY>
</HTML>

```

Sigma maneja el template completo como si fuera un bloque denominado **_global_**. No puede haber dos o más bloques con el mismo nombre. Si un bloque no contiene variables o éstas no fueron asignadas, el bloque completo se omite en la salida. Para modificar este comportamiento, contamos con el método **touchBlock**:

```
$template->touchBlock('nombreBloque');
```

Para realizar la operación inversa, es decir, ocultar un bloque sea cual sea su estado, contamos con el método **hideBlock**, que recibe el nombre del bloque como argumento.

Es posible incluir un bloque dentro de otro. En el siguiente ejemplo, vemos el bloque **ciudades** dentro del bloque **paises**:

- **template.html**

```





```

```

</tr>
<!-- END titulo -->
<!-- BEGIN paises -->
<tr>
    <td valign="top">{nombrePais}</td>
    <td>
        <table border="0" cellspacing="0" cellpadding="0" width="100%" 
            bgcolor="#FFFFCC">
            <!-- BEGIN ciudades -->
            <tr><td>{nombreCiudad}</td></tr>
            <!-- END ciudades -->
        </table>
    </td>
</tr>
<!-- END paises -->
</table>

```

- En **index.php**, primero instanciamos un objeto **HTML_Template_Sigma** y ubicamos el template que se utilizará:

```

require_once 'HTML/Template/Sigma.php';
$tpl = new HTML_Template_Sigma();
$tpl->loadTemplateFile('template.html');

```

Luego definimos la fuente de información para el template, en este caso, un array con países y ciudades:

```

$paises['belgica'][] = 'bruselas';
$paises['japon'][] = 'tokio';

```

III VARIEDAD DE TEMPLATES

Además de Sigma, PEAR ofrece otros motores adicionales de template, como por ejemplo, **Flexy** (**HTML_Template_Flexy**), **IT** (**HTML_Template_IT**), **PHPLIB** (**HTML_Template_PHPLIB**) y **Xipe** (**HTML_Template_Xipe**). Para obtener una completa referencia acerca de cada opción debemos visitar el sitio web oficial del proyecto <http://pear.php.net>.

```
$paises['brasil'][] = 'rio de janeiro';
$paises['japon'][] = 'hiroshima';
$paises['noruega'][] = 'oslo';
$paises['brasil'][] = 'brasilia';

ksort($paises);
```

El bloque **titulo** no contiene variables, por lo que, para mostrarlo, debemos invocar necesariamente el método **touchBlock**:

```
$tpl->touchBlock('titulo');
```

Recorremos el array llenado en el paso anterior, para cargar con valores las variables de cada bloque (**ciudades** y **paises**) utilizando una estructura de repetición, de tipo **foreach**:

```
foreach($paises as $pais => $ciudades) {
    if (!$flag) {
        $paisActual = $pais;
        $flag = true;
        $tpl->setCurrentBlock('paises');
        $tpl->setVariable('nombrePais', $paisActual);
    } elseif ($paisActual != $pais) {
        $paisActual = $pais;
        $tpl->parse('paises');
        $tpl->setCurrentBlock('paises');
        $tpl->setVariable('nombrePais', $paisActual);
    }

    sort($ciudades);

    foreach($ciudades as $ciudad) {
        $tpl->setCurrentBlock('ciudades');
        $tpl->setVariable('nombreCiudad', $ciudad);
        $tpl->parseCurrentBlock();
    }
}
$tpl->parse('paises');
```

Y por último enviamos la salida generada al navegador:

```
$tpl->show();
```

El método **setCurrentBlock** nos permite especificar el bloque actual, es decir, aquel sobre el cual se realizarán las sustituciones. Recibe el nombre del bloque como argumento (de manera predeterminada: **_global**). En el mismo sentido, el método **parseCurrentBlock** parsea el bloque definido anteriormente en **setCurrentBlock**. La salida final del archivo **index.php** es la siguiente:

```
<table border="1" cellspacing="0" cellpadding="5" width="300">
<tr>
    <td valign="top">Paises</td>
    <td valign="top">Ciudades</td>
</tr>
<tr>
    <td valign="top">Belgica</td>
    <td>
        <table border="0" cellspacing="0" cellpadding="0" width="100%">
            bgcolor="#FFFFCC">
            <tr><td>Bruselas</td></tr>
        </table>
    </td>
</tr>
<tr>
    <td valign="top">Brasil</td>
    <td>
        <table border="0" cellspacing="0" cellpadding="0" width="100%">
            bgcolor="#FFFFCC">
            <tr><td>Brasilia</td></tr>
            <tr><td>Rio De Janeiro</td></tr>
        </table>
    </td>
</tr>
<tr>
    <td valign="top">Japon</td>
    <td>
        <table border="0" cellspacing="0" cellpadding="0" width="100%">
            bgcolor="#FFFFCC">
```

```

<tr><td>Hiroshima</td></tr>
<tr><td>Tokio</td></tr>
</table>
</td>
</tr>
<tr>
    <td valign="top">Noruega</td>
    <td>
        <table border="0" cellspacing="0" cellpadding="0" width="100%">
            <tr><td>Oslo</td></tr>
        </table>
    </td>
</tr>
</table>

```

En cualquier momento, podemos conocer el bloque actual mediante una sencilla llamada al método **getCurrentBlock**.

Inclusión de archivos externos

La construcción **INCLUDE** nos permite, justamente, incluir un template desde otro. Su sintaxis requiere que se defina la ruta hacia el archivo que contiene la definición del template que deseamos insertar:

```
<!-- INCLUDE nombreArchivoTemplate -->
```

En el ejemplo incluimos dentro de **template.html** otro template:

- **template.html**

DESARROLLOS PROPIOS

A pesar de la existencia de alternativas disponibles, no es para nada extraño encontrar desarrolladores que programan sus propios motores de template. Ya sea porque no quieren gastar tiempo aprendiendo a utilizar una herramienta nueva o porque ninguna satisface sus necesidades. Por eso, simplemente generan su propia solución.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE> {titulo} </TITLE>
  </HEAD>
  <BODY>

  <!-- INCLUDE contenido.html -->

  </BODY>
</HTML>
```

- **contenido.html**

```
<b>{contenido}<b>
```

- **index.php**

```
<?php

require_once 'HTML/Template/Sigma.php';

$template = new HTML_Template_Sigma('.');
$template->loadTemplatefile('template.html', true, true);
$template->setVariable('titulo', 'Este es el titulo de la pagina');
$template->setVariable('contenido', ucwords('la hora actual es: '))
    .date('h:i A'));
$template->show();

?>
```

Y la salida de **index.php** nos muestra el resultado final (la unión de los respectivos templates) enviado al navegador:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
```

```

<HEAD>
    <TITLE> Este es el titulo de la pagina </TITLE>
</HEAD>
<BODY>

    <b>La Hora Actual Es: 10:37 PM</b>

</BODY>

</HTML>

```

Funciones

Podemos incluir funciones dentro del template anteponiendo al nombre de la función el prefijo **func_** y ubicando los argumentos (si es que los hay) entre paréntesis. Luego, con el método **setCallbackFunction**, es posible ligar la función del template a una escrita en PHP, como vemos en el siguiente código:

```

<?php

require_once 'HTML/Template/Sigma.php';
$tpl = new HTML_Template_Sigma();
$tpl->loadTemplateFile('template.html');

function formatoMoneda($valor) {
    return '$ '.number_format($valor, 2, ',', '.');
}

$tpl->setCallbackFunction('moneda', 'formatoMoneda');
$tpl->setVariable('precio', 123);

$tpl->show();

?>

```

- **template.html**

```
El costo es de func_moneda('{precio}')
```

Si ejecutáramos el ejemplo anterior, obtendríamos como salida:

```
El costo es de $ 123,00
```

Otros métodos disponibles

Sigma mantiene un gran número de métodos que intentan resolver las necesidades de los desarrolladores. Además de los ya vistos, podemos citar:

MÉTODO	DESCRIPCIÓN
addBlock	Permite incluir un nuevo bloque en el lugar ocupado por una variable (primer argumento). El segundo argumento es el nombre del nuevo bloque y el tercero, su contenido. El método addBlockfile es similar, sólo que el contenido se obtiene desde un archivo externo (tercer argumento).
blockExists	Recibe un nombre de bloque y verifica si existe o no.
getBlockList	Devuelve un array con los bloques contenidos. Recibe como argumentos un nombre de bloque (el parent) y si la búsqueda se realizará más allá del primer nivel.
getPlaceholderList	Similar a getBlockList , sólo que recibe un único argumento, el nombre del bloque.
placeholderExists	Recibe un nombre de variable y verifica si existe o no (puede buscar sólo en el bloque dado como segundo argumento).
replaceBlock	Reemplaza el contenido de un bloque (primer argumento) por otro (segundo argumento). Con replaceBlockfile , podemos obtener el contenido desde un archivo externo (segundo argumento).

Tabla 1. Métodos disponibles en Sigma.

Ejemplo completo de uso

Para ver cómo funciona Sigma en un ejemplo real, tomemos el caso de un sitio cuyas páginas están divididas en templates. Así cada una se compondrá por:

- 1) Un encabezado (`include_encabezado.html`)

```
<!-- BEGIN css -->
<link rel="stylesheet" href="css/styles.css" type="text/css">
<!-- END css -->

<table width="780" cellspacing="0" cellpadding="0" border="0" align="center">
  <tr>
    <td style="padding-left:25px" bgcolor="#5678BB" class="logo" height="75"
```

```

background="images/bg.jpg"> {tituloEncabezado}</td>
</tr>
<tr>
<td height="22" colspan="2" bgcolor="#999999">
<table width="100%" cellspacing="0" cellpadding="0" border="0">
<tr>
<td height="22" bgcolor="#AEC3E5" class="text" align="center">

<a href="./">Home</a>
| <a href=".//historia">Historia</a>
| <a href=".//noticias">Noticias</a>
| <a href=".//productos">Productos</a>
| <a href=".//contacto">Contacto</a>

</td>
</tr>
</table>
</td>
</tr>
</table>

```

2) Un menú a la izquierda (**include_menu.html**)

```

<!-- BEGIN css -->
<link rel="stylesheet" href="css/styles.css" type="text/css">
<!-- END css -->

<table width="180" cellspacing="0" cellpadding="8" border="0"
bgcolor="#CCCCCC" height="100%">
<tr>
<td class="text" valign="top">
<p><b>Opciones</b>
<br>

<br> :: <a href="./">Home</a>
<br> :: <a href=".//historia">Historia</a>
<br> :: <a href=".//noticias">Noticias</a>
<br> :: <a href=".//productos">Productos</a>
<br> :: <a href=".//contacto">Contacto</a>

```

```

</p>
</td>
</tr>
</table>

```

3) Un contenido principal (**template_index.html**, **template_contacto.html**, etcétera)

```

<!-- BEGIN css -->
<link rel="stylesheet" href="css/styles.css" type="text/css">
<!-- END css -->

<table width="600" cellspacing="10" cellpadding="0" align="center">
    border="0" bgcolor="#eeeeee">
<tr>
    <td valign="top" class="text">
        <p><span class="title">Esta es la pagina de inicio!</span></p>
        <p> Lorem ipsum dolor sit amet, ... </p>
    </td>
</tr>
</table>

```

Figura 5. Sigma nos permite estructurar proyectos de manera intuitiva.

4) Un pie de página (**include_pie.html**)

```

<!-- BEGIN css -->
<link rel="stylesheet" href="css/styles.css" type="text/css">

<!-- END css -->

<table width="780" cellspacing="0" cellpadding="0" height="40" border="0"
       align="center">
  <tr>
    <td bgcolor="#AEC3E5" height="24" class="text" align="center">
      Copyright &copy;
      2008 - Lorem ipsum dolor sit amet, consetetur sadipscing elitr,
      sed diam nonumy.</td>
  </tr>
</table>

```

5) El template **include_base.html** reúne las partes componentes de cada página:

```

<html>
  <head>
    <title>{tituloPagina}</title>
    <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
    <link rel="stylesheet" href="css/styles.css" type="text/css">
  </head>

  <body bgcolor="#999999" leftmargin="0" topmargin="0" marginwidth="0"
        marginheight="0">

    {include_encabezado}

    <table width="780" cellspacing="0" cellpadding="0" border="0" align="center">
      <tr>
        <td valign="top" width="180" height="100%">
          {include_menu}
        </td>
      </tr>
    </table>
  </body>
</html>

```

```

<td valign="top" width="*">
{include_contenido}

</td>

</tr>

</table>
{include_pie}
</body>
</html>

```

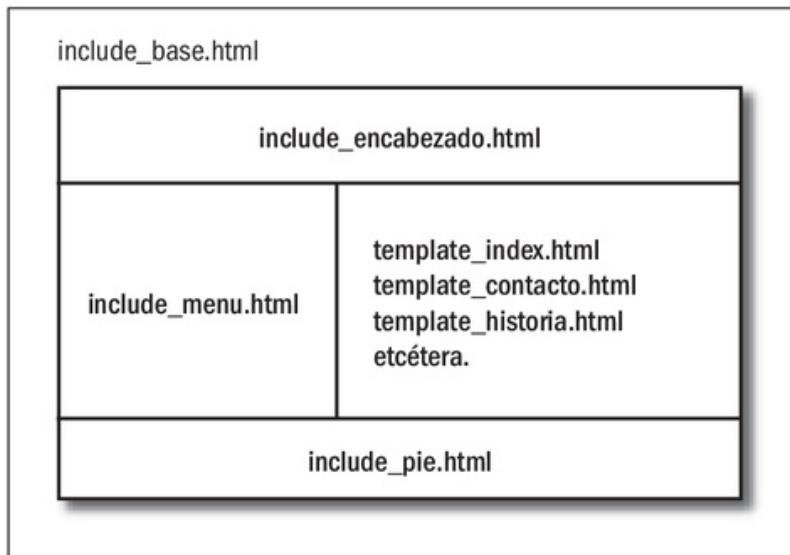


Figura 6. El uso de templates nos permite modularizar partes de una página según su función o posición dentro de la estructura general.

Ubicaremos los templates preparados en la carpeta **cache**. Cada vez que se intente ingresar en una dirección, mediante un archivo **.htaccess** se redireccionará al **index.php**. El contenido del archivo **.htaccess** es el siguiente:

III RAD

El desarrollo rápido de aplicaciones con el lenguaje PHP es una técnica que cuenta cada vez con más adeptos, y una de sus cualidades es la utilización de templates. Veremos una referencia acerca de este tema en el **Capítulo 4** de este libro, donde trataremos este aspecto, tomando como centro al framework **CakePHP**.

ErrorDocument 404 /sigma/index.php

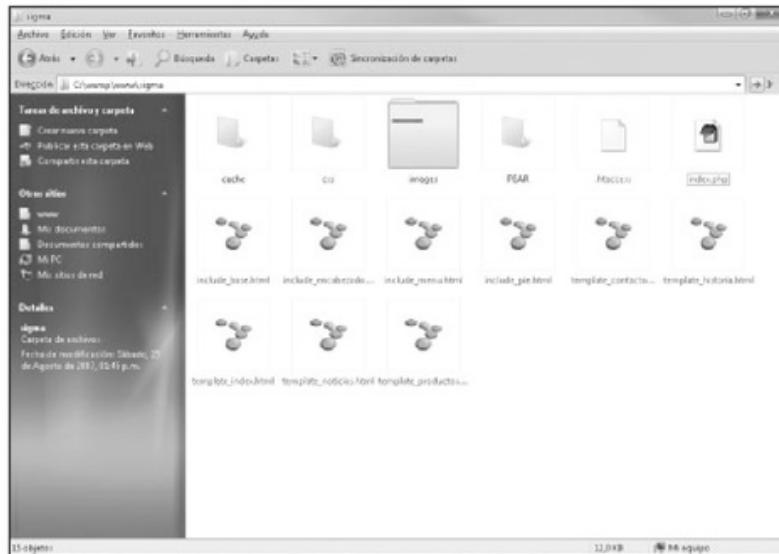


Figura 7. A través de Sigma, es posible facilitar y equilibrar las tareas de los desarrolladores y diseñadores.

Ya en **index.php**, ligaremos el tercer argumento de la variable **REQUEST_URI** (la ruta después del **root**) para cargar el template correspondiente:

```
header("HTTP/1.1 200 OK");

$uri = explode("/", $_SERVER[REQUEST_URI]);
$uri = $uri[2] ? $uri[2] : 'index';
if (!file_exists($pagina = 'template_.'.$uri.'.html')) {
    echo '<h1>Pagina no encontrada !</h1>';
    exit;
}
```

Si, por ejemplo, ingresáramos la dirección desde un navegador, internamente se invocaría desde el **index.php** al template **template_historia.html**:

III MVC

MVC (*Model View Controller*, Modelo Vista Controlador) es un modelo muy popular y ampliamente utilizado por quienes realizan labores realacionadas con el diseño de sistemas. Es importante saber que principalmente avanza sobre el tema de la separación en capas de una aplicación, algo ligado directamente a la utilización de templates.

<http://localhost/sigma/historia>

Para seguir con el contenido del **index.php**, requerimos la clase base de Sigma y cargamos las variables con los templates correspondientes:

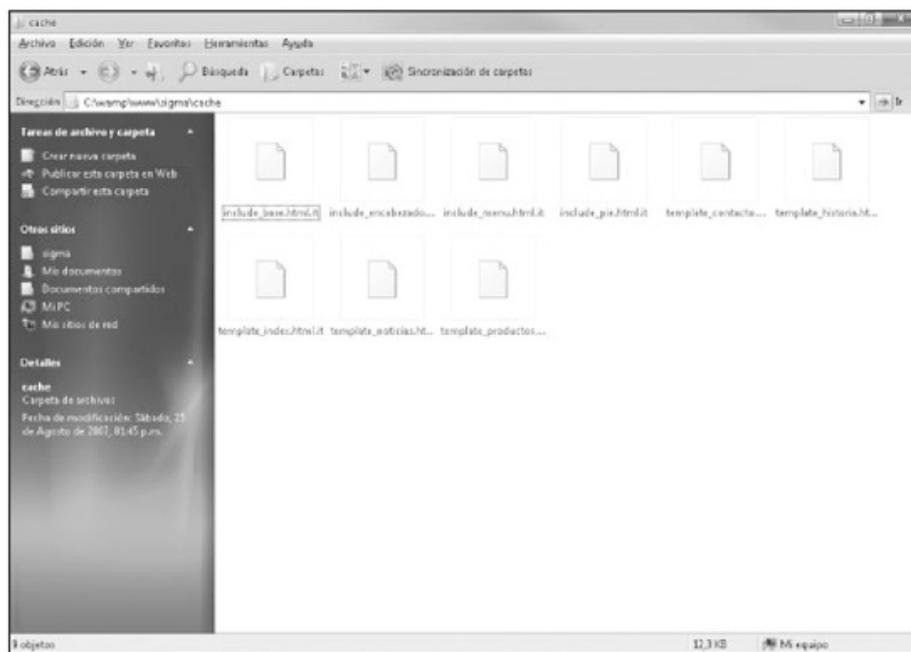


Figura 8. La caché es transparente al desarrollador y nos permite agilizar la carga de los templates.

```
$tplInclude =& new HTML_Template_Sigma('..', './cache');

$tplInclude->loadTemplatefile('include_encabezado.html', true, true);
$tplInclude->setVariable('tituloEncabezado', 'Pagina actual: '.$uri);
$tplInclude->parse();
$tpl->setVariable('include_encabezado', $tplInclude->get());
$tplInclude->loadTemplatefile('include_menu.html', true, true);
$tplInclude->parse();
```

III MÁS SOBRE PEAR

PEAR, en su web oficial que podemos encontrar en la dirección <http://pear.php.net/>, mantiene una gran cantidad de paquetes orientados a diversos temas, por ejemplo la implementación de templates. Conoceremos más acerca de este importante proyecto (características, instalación y forma de uso) en el **Capítulo 8** de este libro.

```
$tpl->setVariable('include_menu', $tplInclude->get());
$tplInclude->loadTemplatefile($pagina, true, true);
$tplInclude->parse();
$tpl->setVariable('include_contenido', $tplInclude->get());
$tplInclude->loadTemplatefile('include_pie.html', true, true);
$tplInclude->parse();
$tpl->setVariable('include_pie', $tplInclude->get());
$tpl->setVariable('tituloPagina', 'Bienvenido ! - '.date("d.m.Y / h:i A"));
$tpl->show();
```

En el ejemplo, se supone que el sitio está contenido en un directorio llamado **sigma**, que pende del **documentRoot** del servidor. Si en nuestro caso no ocurre de esta manera, deberemos modificar los archivos **.htaccess** e **index.php** según corresponda. El siguiente fragmento no aparece en el código que se genera, pero nos puede servir para editar de una manera amigable los templates por separado:

```
<!-- BEGIN css -->
<link rel="stylesheet" href="css/styles.css" type="text/css">
<!-- END css -->
```

Esta es la pagina de inicio!

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consetetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat.

Figura 9. Los templates generados se podrán manipular de forma independiente.

La fuente del template que utilizamos en el proyecto es www.templateyes.com.

SMARTY

La función básica y necesaria para un motor de templates reside en separar la lógica del diseño en una aplicación. Smarty, por supuesto, nos otorga la posibilidad de llevar a cabo esta acción, pero va mucho mas allá, porque incluye un avanzado sistema de caché, estructuras de control, extensiones, y opciones para interferir el diseño de las páginas. Además, introduce el concepto de templates compilados, que son scripts PHP generados a partir de templates. La invocación se realiza sobre el archivo compilado, que se vuelve a compilar cuando el documento original cambia.

The screenshot shows the Smarty Template Engine website. At the top, there's a navigation bar with links for download, documentation, faq, forum, mailing lists, changelog, and contribs. Below the navigation is a search bar. The main content area features a banner for "Spa -70 %" offers. It highlights the release of "Smarty 3.0 RC4 Released" (Oct 5, 2010), noting it's the fourth release candidate. It encourages users to join the developer mailing list and see the Smarty 3 section of the forums. Documentation is still being finalized. Below this, there's information about "Smarty 3.0 RC3 Released" (Jul 14, 2010) and "Smarty 3.0 RC2 Released" (Jun 14, 2010). To the right, there's a sidebar for "Stock photos for \$1" featuring a large "\$1" and a list of categories: Business & fun illustrations, Food & diets, Celebrity & people, Sport & racing, Beauty & lifestyle, Nature & animals, Christmas, Outdoor, Office, and And a lot more!

Figura 10. Smarty es una herramienta profesional de uso libre y código fuente abierto.

Podemos descargar Smarty desde <http://smarty.net>.

Modos de trabajo

Las funcionalidades ofrecidas por Smarty están segmentadas en dos partes: una orientada a los diseñadores y otra para programadores.

Los críticos de esta herramienta sostienen que Smarty complejiza y ensucia innecesariamente la función inicial de un motor de templates, convirtiéndose casi en un lenguaje de programación por encima de PHP.

Por otra parte, las empresas profesionales cuyos miembros trabajan desde hace ya un tiempo con Smarty no ven complejidad, sino una respuesta a las necesidades de cada caso. Para utilizar esta herramienta, debemos solamente copiar el directorio **libs** (que

viene junto con la distribución) a un lugar accesible por las páginas PHP. Luego, será necesario instanciar la clase **Smarty** como veremos en los siguientes ejemplos:

```
<?php  
  
require 'Smarty.class.php';  
$smarty = new Smarty;  
  
?>
```

Una opción válida es definir la constante **SMARTY_DIR**, asignándole el directorio en donde se ubica la clase principal:

```
define('SMARTY_DIR', 'c:/www/proyecto/smarty/lib/');  
  
define('SMARTY_DIR', '/usr/local/lib/php/Smarty/libs/');  
  
<?php  
  
require SMARTY_DIR.'Smarty.class.php';  
$smarty = new Smarty;  
  
?>
```

Una aplicación Smarty se estructura básicamente mediante cuatro directorios, cada uno de los cuales cumple una función específica:

- **template_dir** (aquí ubicamos los templates)
- **compile_dir** (templates compilados)
- **config_dir** (archivos de configuración)
- **cache_dir** (caché de templates)

Para definir estos directorios en la página usamos las propiedades del objeto **Smarty**:

```
$smarty->template_dir = './smarty/templates';  
$smarty->compile_dir = './smarty/templates_c';
```

```
$smarty->config_dir = './smarty/configs';
$smarty->cache_dir = './smarty/cache';
```

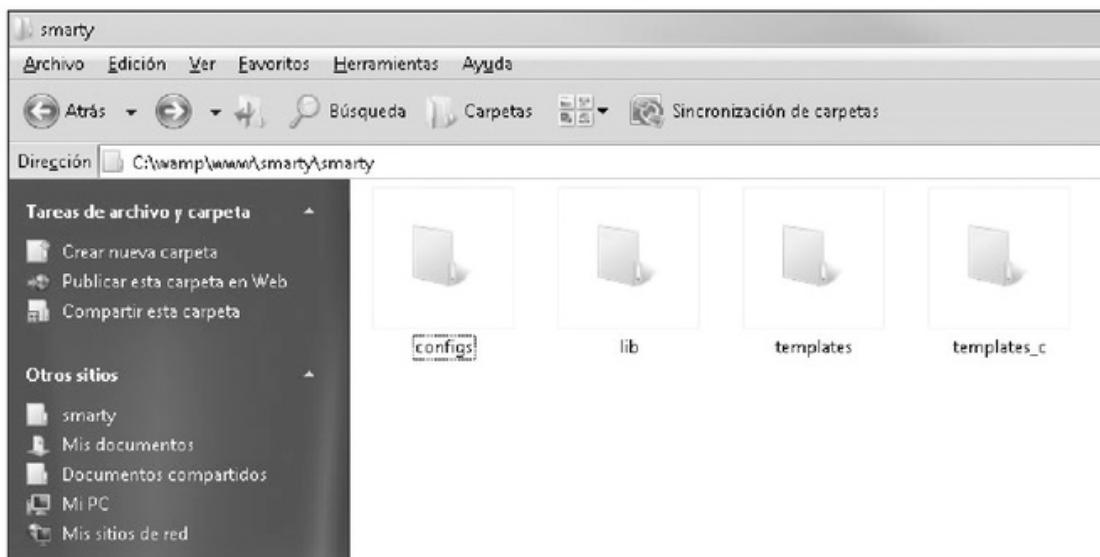


Figura 11. Smarty estructura su funcionalidad de manera clara a través de un árbol de directorios.

Por cuestiones de seguridad, recomendamos dejar estas carpetas fuera del directorio raíz del sitio, para evitar la posibilidad de que puedan ser accedidas en forma directa vía web. De esta manera se vuelve completamente imposible el acceso (a través de código malicioso) a nuestros templates.

Variables

Las variables pueden ser simples, arrays, propiedades, métodos de objetos, etcétera. Los delimitadores por defecto son las llaves de apertura ({}) y de cierre (}):

```
{$nombreVariable}
```

Podemos modificar este comportamiento utilizando las propiedades especiales **left_delimiter** y **right_delimiter**:

```
$smarty->left_delimiter = '<<';
$smarty->right_delimiter = '>>';
```

```
<<$variable>>
```

Dentro de un script, es posible cargar las variables con valores mediante el método **assign** mostrado a continuación:

```
<?php  
  
$smarty->assign('variable1', 'valor1');  
$smarty->assign('variable2', 'valor2');  
  
?>
```

```
<p>{$variable1}, <b>{$variable2}</b></p>
```

Son sensibles a mayúsculas y minúsculas, y pueden incluirse entre comillas dobles. Los arrays de PHP suelen ser accedidos por variables Smarty desde los templates, como veremos a continuación el código es similar al necesario para accederlos desde PHP. La sintaxis utilizada para arrays asociativos y numéricos es:

```
$variable.nombreClave
```

```
$variable[pos1][pos2][posN]
```

Sucede lo mismo con los objetos:

```
$variable->nombreAtributo
```

Podemos capturar (mediante el método **capture**) un bloque del template y almacenarlo en una variable para mostrarlo luego:



DECISIÓN

Hay quienes piensan que incluir las funcionalidades brindadas por Smarty en un template liga de manera definitiva el diseño a la herramienta, y que también demanda a los diseñadores apartarse del lenguaje HTML para conocer a fondo la sintaxis de Smarty. Esto, generalmente, se resuelve a partir de las necesidades y los conocimientos del personal.

```
{capture name=nombreCaptura}
Esta es la captura
{/capture}

{if $smarty.capture.nombreCaptura ne ""}
<b>{$smarty.capture.nombreCaptura}</b>
{/if}
```

Completaremos la información sobre estructuras de control más adelante. Smarty nos provee funciones para aplicar sobre variables directamente en el template. Algunas de ellas son:

- **capitalize**
- **cat**
- **count_characters**
- **count_paragraphs**
- **count_sentences**
- **count_words**
- **date_format**
- **default**
- **escape**
- **indent**
- **lower**
- **nl2br**
- **regex_replace**
- **replace**
- **spacify**
- **string_format**
- **strip**
- **strip_tags**
- **truncate**
- **upper**
- **wordwrap**

Para utilizarlas, disponemos de la siguiente sintaxis:

```
{$nombreVariable|nombreFuncion:argumento1:argumento2:argumentoN}
{$smarty.now|date_format:"%Y/%m/%d"}
{$nombreVariable|upper}
```

El símbolo | (**pipe**) indica que la salida del primer término se envía al segundo, y así sucesivamente, por lo que algo como lo siguiente es válido:

```
{$nombreVariable|nl2br|upper}
```

The screenshot shows the Smarty Debug Console interface in a browser window. It displays several sections of data:

- Included templates & config files (load time in seconds):**
 - idiomas.conf (0.00226) global
 - include_base.tpl (0.00159)
 - include_encabezado.tpl (0.00085)
 - include_pquierda.tpl (0.00024)
 - include_contenido.tpl (0.00222)
 - include_pquierda2.tpl (0.0024)
- assigned template variables:**

```
(SCRIPT_NAME)
($idIdiomaSelected)
($rbOpcionesIdioma)
```
- assigned config file variables (outer template scope):**

```
(#filed)
(#var#)
```

	Value
idiomas.conf	true
Array (14)	
tituloSeleccionIdioma	> "Seleccione su idioma"
tituloInicioSession	> "Inicio de Sesión"
tituloRegina	> "Example Smarty"
mensajeSesion	> "Bienvenido al Website de la Compañia"
mensajeLogin	> "Por favor ingrese su nombre de usuario..."
errorLogin	> "Error en registros!"
mensajeBase	> "Inicio"
mensajeCompany	> "Compañia"
mensajeServicios	> "Servicios"
mensajeProductos	> "Productos"
mensajeClients	> "Clientes"
mensajeDownloads	> "Descargas"
mensajeSiteMap	> "Mapa del Sitio"
mensajeContactUs	> "Contacto"

Figura 12. Smarty nos permite trabajar en modo debug, algo importante para calcular el rendimiento de las aplicaciones.

Comentarios

Los comentarios se definen entre los delimitadores {* y *}.

```
{* esto es un comentario *}
```

Archivos de configuración

En estos archivos, Smarty nos possibilita guardar variables y asignarles valores. Luego podremos incluirlos en un template y acceder a las variables contenidas. Un archivo de configuración tiene la forma:

III DOCUMENTACIÓN

Uno de los puntos fuertes de Smarty es su extensa documentación, la cual abarca desde los aspectos más generales a los más puntuales, incluyendo consejos prácticos de uso y ejemplos reales de implementación de soluciones. Cada nueva versión de esta herramienta provoca una revisión del manual y su actualización inmediata.

```
nombreVariable1 = "valor1"
nombreVariable2 = "valor2"
nombreVariableN = "valorN"
```

Mediante **config_load**, podremos incorporar el archivo y utilizar las variables a través de cualquiera de las siguientes maneras:

```
{#nombreVariable#}
{$smarty.config.nombreVariable}:
```

Por ejemplo:

```
{config_load file="nombreArchivo.conf"}
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> {$smarty.config.nombreVariable1} </TITLE>
</HEAD>
<BODY>
{#nombreVariable2#}
</BODY>
</HTML>
```

Es posible segmentar cada archivo de configuración en secciones más pequeñas delimitadas por nombres entre corchetes:

```
nombreVariable1 = "valor1"
```

{ } VARIEDAD DE FUNCIONES

Es importante saber que Smarty es capaz de proveernos funcionalidades tan diversas como generar popups, incluir ecuaciones matemáticas y dar formato al texto, entre muchas otras. Por esta razón queda a criterio de cada uno de los diseñadores la incorporación de estas opciones avanzadas dentro de los archivos que contienen los templates.

```

nombreVariable2 = "valor2"
nombreVariableN = "valorN"

[nombreSeccion1]
nombreVariableN = "valorN"
[nombreSeccion2]
nombreVariableN = "valorN"

```

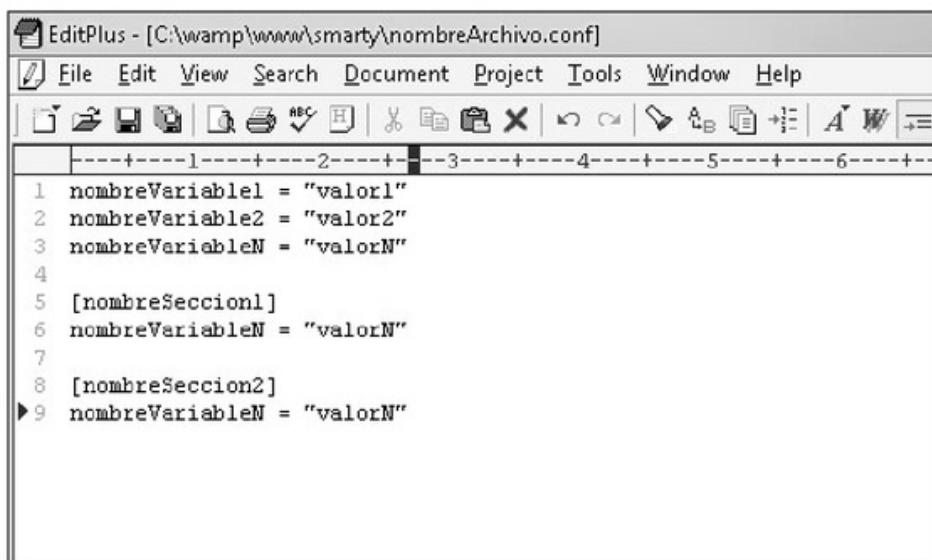
Una sección que comienza con un punto permanecerá oculta, no accesible desde los templates que la incorporan.

Las variables que no están por debajo de ningún bloque son globales. Para cargar una sección en particular, utilizamos el atributo **section**:

```
{config_load file="nombreArchivo.conf" section="nombreSeccion1"}
```

También podemos incluir archivos de configuración desde el código de la aplicación. El nombre de la sección es opcional:

```
$smarty->config_load('nombreArchivo.conf', 'nombreSeccion1');
```



The screenshot shows a window titled 'EditPlus - [C:\wamp\www\smarty\nombreArchivo.conf]'. The menu bar includes File, Edit, View, Search, Document, Project, Tools, Window, Help. The toolbar has icons for Open, Save, Print, Find, Replace, and others. The main text area contains the following code:

```

1 nombreVariable1 = "valor1"
2 nombreVariable2 = "valor2"
3 nombreVariableN = "valorN"
4
5 [nombreSeccion1]
6 nombreVariableN = "valorN"
7
8 [nombreSeccion2]
9 nombreVariableN = "valorN"

```

Figura 13. Trabajar con archivos de configuración facilita nuestros desarrollos a la vez que descentraliza la ubicación de la información.

La variable especial **smarty** tiene muchos otros usos, entre los que se cuentan:

- Acceder a los valores de los arrays especiales de PHP:

```
$smarty.nombreArray.nombreClave  
{*  
nombreArray puede ser:  
get,  
post,  
request,  
cookies,  
server,  
env o  
session  
*}
```

- Obtener el **timestamp** actual:

```
$smarty.now
```

- Acceder al valor de una constante:

```
$smarty.const.NOMBRECONSTANTE
```

- Obtener el nombre del template actual:

```
$smarty.template
```

- Obtener la versión de Smarty:

```
$smarty.version
```

III FORMATO DE TEXTO

La función **textformat** permite dar formato a un texto dentro del template. Tiene multitud de opciones (indentación, estilos, límite de líneas, etcéteral) y puede ser de utilidad para el diseño de sitios. Recordemos que podemos emular la funcionalidad de este tipo de opciones desde la página PHP que genera los contenidos del template.

Propiedades y métodos del objeto Smarty

Además de las funciones ya mencionadas, podremos utilizar un objeto **Smarty** con las siguientes propiedades para modificar algún comportamiento específico de nuestros templates:

PROPIEDAD	DESCRIPCIÓN
autoload_filters	Autocarga de filtros. Recibe como argumento un array asociativo (las claves serán los tipos de filtro, y los valores, los nombres de cada uno).
cache_handler_func	Función que asume el control de la caché.
cache_lifetime	Duración (en segundos) que una caché de template es válida (-1 para nunca expirar).
cache_modified_check	Imprime en la página si el template es cacheado o fue regenerado.
caching	Permite a Smarty conocer o no la existencia de una versión del template en la caché.
compile_check	Verifica o no si los templates fueron modificados, para decidir si se vuelven a compilar.
compile_id	Define un identificador para el compilador. Cada compilador puede acceder a un único directorio de templates (template_dirs).
config_booleanize	Convierte los valores de las variables de configuración en booleanos, para luego poder utilizarlos en expresiones.
config_overwrite	Si hay más de una variable con el mismo nombre en la misma sección de un archivo de configuración y, si config_overwrite es falso, las variables se tratan como un array, en caso contrario, se toma el valor asignado a la última.
config_read_hidden	Si es verdadero, oculta las secciones que comienzan con un punto.
debugging	Define si se trabaja en modo debug o no.
default_template_handler_func	Función que se llamará cuando un template no se encuentre.
error_reporting	Define el valor de la directiva error_reporting de PHP.
force_compile	Obliga a la recompilación de templates.
plugins_dir	Directorio de ubicación de los plug-ins.
request_use_auto_globals	Define si se van a utilizar variables globales de PHP o no.
secure_dir	Origen único de los templates (sólo si security es verdadero).
security	Configura ciertas medidas de seguridad (ver secure_dir y security_settings).
security_settings	Admite las constantes IF_FUNCS (es un array de funciones PHP permitidas en bloques IF), INCLUDE_ANY (incluir o no un template desde cualquier directorio), PHP_TAGS (admite o no etiquetas {php}{/php}), MODIFIER_FUNCS (funciones PHP admitidas para modificar variables), ALLOW_CONSTANTS (admite o no las constantes \$smarty.const.nombreConstante). Sólo si security es verdadero.
trusted_dir	Array de directorios confiables (sólo si security es verdadero).
use_sub_dirs	Permite crear o no subdirectorios en compile_dir y cache_dir .

Tabla 2. Propiedades del objeto Smarty.

Y entre los métodos podemos incluir los siguientes:

MÉTODO	DESCRIPCIÓN
append	Recibe como argumentos un nombre de variable y un valor asignado. Podemos realizar sucesivas llamadas a este método sobre la misma variable.
append_by_ref	Similar al anterior, sólo que pasa valores por referencia.
assign_by_ref	Asigna valores por referencia (similar a assing).
clear_all_assign	Limpia el valor de las variables asignadas con anterioridad.
clear_all_cache	Elimina la caché del template.
clear_assign	Limpia el valor de la variable pasada como argumento.
clear_cache	Elimina el caché del template pasado como argumento.
clear_compiled_tpl	Elimina la compilación del template pasado como argumento.
clear_config	Elimina las variables de configuración.
get_config_vars	Recibe una variable de configuración y retorna su valor.
get_registered_object	Recibe un objeto asignado y devuelve una referencia.
get_template_vars	Recibe una variable y retorna su valor.
is_cached	Devuelve verdadero si el template pasado como argumento está en caché.
register_object	Registra un objeto para utilizar dentro del template.
template_exists	Devuelve verdadero si el template pasado como argumento existe.
trigger_error	Muestra un mensaje de error. El nivel es pasado como argumento (de manera predeterminada: E_USER_WARNING).
unregister_object	Desregistra un objeto.

Tabla 3. Métodos del objeto Smarty.

Operadores

Los siguientes son operadores validos en Smarty:

- Igual: **==, eq**
- No igual: **!=, ne, neq**
- Mayor que: **>, gt**
- Menor que: **<, lt**
- Mayor o igual que: **>=, gte**
- Menor o igual que: **<=, lte**
- Idéntico a: **==**
- Negación: **!, not**
- Módulo: **%, mod**

La disponibilidad de operadores que es posible incluir dentro de templates resulta otra expresión de las capacidades extras que Smarty pone a disposición tanto del diseñador como del programador, y que, seguro, van más allá de un simple formato.

Condicionales

Las estructuras condicionales son de gran utilidad dentro de los templates para administrar la exhibición de determinadas partes de un documento según el valor actual de una variable. La incorporación se realiza de la siguiente manera:

```
{if $variable eq "valor1"}  
...  
{elseif $variable eq "valor2"}  
...  
{else}  
...  
{/if}
```

Ciclos

Con `foreach`, es posible iterar un array PHP:

```
$paises = array ('Brasil', 'Noruega', 'Francia');  
asort($paises);  
$smarty->assign('paises', $paises);  
$smarty->display('index.tpl');
```



Figura 14. Para trabajar con Smarty, debemos incluir el archivo que contiene la clase base y generar la estructura de directorios necesaria.

- index.tpl

```
{foreach from=$paises key=idPais item=nombrePais}  
    <li>nombre del pais: {$nombrePais}
```

```
(posicion {$idPais})
{/foreach}
```

A través de la variable especial **smarty**, obtenemos información acerca de la iteración:

```
$smarty.foreach.nombreForeach.nombrePropiedad
```

Donde **nombrePropiedad** puede ser:

- **iteration**: iteración actual.
- **first: true** si es la primera iteración.
- **last: true** si es la última iteración.
- **total**: nos muestra el número de iteraciones del **foreach**.

Las secciones también pueden ser iteradas a través de los atributos **loop** (nombre de la variable), **start** (índice de comienzo), **step** (salta del **start** al **start+step**, -1 para ir desde el fin al comienzo), y **max** (máximo número de iteraciones):

```
{section name=pais loop=$paises start=1}
<li>nombre del pais: {$paises[pais]}
(posicion {$smarty.section.pais.index},
 iteracion nro {$smarty.section.pais.iteration})
{sectionelse}
    No hay paises disponibles !
{/section}
```

Como vimos, es posible obtener información acerca de la iteración, en la sintaxis:

```
$smarty.section.nombreSeccion.nombrePropiedad
```

Donde **nombrePropiedad** puede ser:

- **index**: índice actual.
- **index_prev**: índice anterior.
- **index_next**: siguiente índice.
- **iteration o rownum**: iteración actual.
- **first: true** si es la primera iteración.

- **last**: **true** si es la última iteración.
- **total**: nos muestra el número de iteraciones del **foreach**.
- **loop**: último número del índice.

Inclusión de archivos externos

Dentro de un template, podemos recuperar el contenido de otros archivos, lo que permite modularizar los diseños:

```
{include file="template.tpl"}  
{include file="template.tpl" assign="nombreVariable"}  
{* para redirigir el contenido a una variable *}
```

Como alternativa existe la posibilidad de utilizar **fetch**:

```
{fetch file="nombreArchivo"}
```

Funciones

Los argumentos pasados a las funciones son similares a los atributos de los elementos HTML. Para generar elementos **checkbox**, contamos con **html_checkboxes**:

```
<?php  
  
require 'smarty/lib/Smarty.class.php';  
$smarty = new Smarty;  
  
$smarty->template_dir = './smarty/templates';  
$smarty->compile_dir = './smarty/templates_c';  
$smarty->config_dir = './smarty/configs';  
$smarty->cache_dir = './smarty/cache';  
  
for ($c=1;$c<=10;$c++)  
    $opciones[$c] = "Valor $c";  
  
$smarty->assign('opciones', $opciones);  
$smarty->assign('idOpcionSeleccionada', array(3, 4, 5, 10));  
$smarty->display('index.tpl');
```

?>

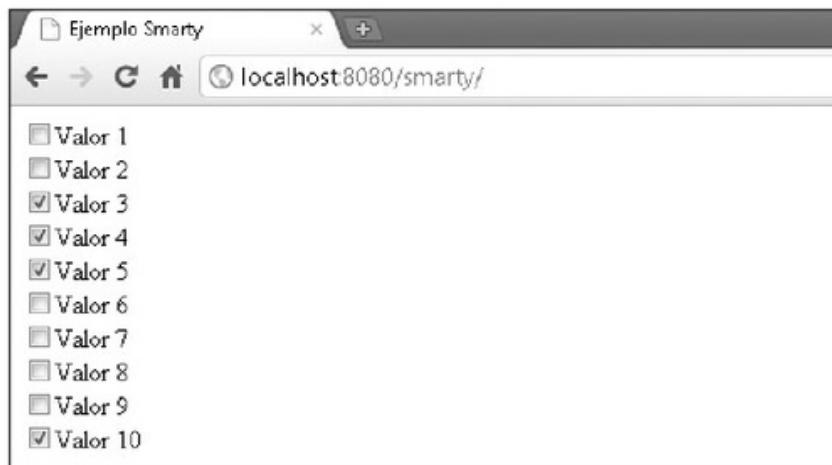


Figura 15. Los diseñadores tienen con Smarty la posibilidad de incluir elementos HTML de una manera alternativa.

- **index.tpl:**

```
{html_checkboxes name="idOpcion" options=$opciones
selected=$idOpcionSeleccionada separator="  
"}
```

Salida:

```
<label><input type="checkbox" name="idOpcion[]" value="1" />Valor
1</label><br />
<label><input type="checkbox" name="idOpcion[]" value="2" />Valor
2</label><br />
<label><input type="checkbox" name="idOpcion[]" value="3" checked="checked"
/>Valor 3</label><br />
<label><input type="checkbox" name="idOpcion[]" value="4" checked="checked"
/>Valor 4</label><br />
<label><input type="checkbox" name="idOpcion[]" value="5" checked="checked"
/>Valor 5</label><br />
<label><input type="checkbox" name="idOpcion[]" value="6" />Valor
6</label><br />
<label><input type="checkbox" name="idOpcion[]" value="7" />Valor
7</label><br />
<label><input type="checkbox" name="idOpcion[]" value="8" />Valor
8</label><br />
```

```
<label><input type="checkbox" name="idOpcion[]" value="9" />Valor  
9</label><br />  
<label><input type="checkbox" name="idOpcion[]" value="10"  
checked="checked" />Valor 10</label><br />
```

Para generar elementos **img**, contamos con **html_image**:

```
<?php  
  
require 'smarty/lib/Smarty.class.php';  
$smarty = new Smarty;  
  
$smarty->template_dir = './smarty/templates';  
$smarty->compile_dir = './smarty/templates_c';  
$smarty->config_dir = './smarty/configs';  
$smarty->cache_dir = './smarty/cache';  
  
$smarty->assign('archivo', 'http://www.google.com.ar/intl/en_com/images/  
logo_plain.png');  
$smarty->assign('alto', 110);  
$smarty->assign('ancho', 276);  
  
$smarty->assign('alt', 'Google');  
$smarty->assign('enlace', 'http://www.google.com.ar/');  
  
$smarty->display('index.tpl');  
  
?>
```

- **index.tpl:**

III MODOS DE SMARTY

Smarty está disponible en dos modos: para programación de templates y para diseño de templates. Hay quienes prefieren utilizar sólo la referida a la programación, dejando que los diseñadores resuelvan su parte según sus gustos. Esto no afecta de ningún modo el correcto funcionamiento de la herramienta ni limita su potencial.

```
{html_image file="$archivo" height="$alto" width="$ancho" alt="$alt"
    href="$enlace" border="0"}
```



Figura 16. Es posible incluir imágenes en un template a través de Smarty.

Salida:

```
<a href="http://www.google.com.ar/"></a>
```

Para generar opciones de elementos **select**, contamos con **html_options**:

```
<?php

require 'smarty/lib/Smarty.class.php';
$smarty = new Smarty;

$smarty->template_dir = './smarty/templates';
$smarty->compile_dir = './smarty/templates_c';
$smarty->config_dir = './smarty/configs';
$smarty->cache_dir = './smarty/cache';

for ($c=1;$c<=10; $c++) {
    $textos[$c] = "Valor $c";
    $ids[$c] = $c*10;
}

$smarty->assign('id', array_values($ids));
$smarty->assign('texto', array_values($textos));
$smarty->assign('idSeleccionado', 4*10);
```

```
$smarty->display('index.tpl');

?>
```

- index.tpl

```
<select name=nombreSelect>
{html_options values=$id output=$texto selected=$idSeleccionado}
</select>
```

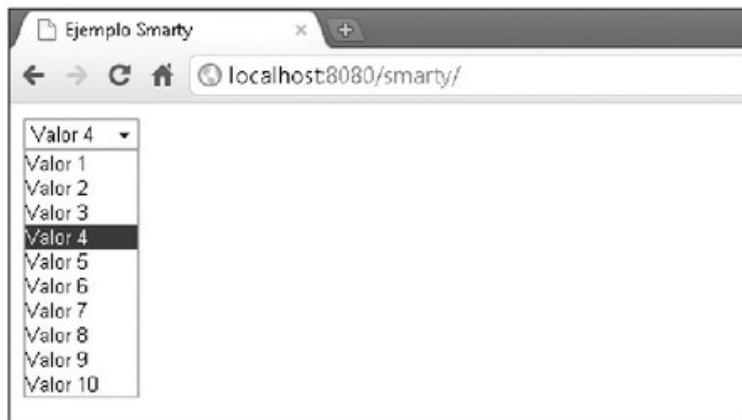


Figura 17. Smarty nos posibilita generar elementos HTML desde un template.

Salida:

```
<select name="nombreSelect" id="nombreSelect">
<option label="Valor 1" value="10">Valor 1</option>
<option label="Valor 2" value="20">Valor 2</option>
<option label="Valor 3" value="30">Valor 3</option>
<option label="Valor 4" value="40" selected="selected">Valor 4</option>
<option label="Valor 5" value="50">Valor 5</option>
<option label="Valor 6" value="60">Valor 6</option>
<option label="Valor 7" value="70">Valor 7</option>
<option label="Valor 8" value="80">Valor 8</option>
<option label="Valor 9" value="90">Valor 9</option>
<option label="Valor 10" value="100">Valor 10</option>
</select>
```

Si las opciones y los valores están en el mismo array asociativo, podemos utilizar el atributo **options**:

```

for ($c=1;$c<=10;$c++) {
    $opciones[$c*10] = "Valor $c";
}

$smarty->assign('opciones', $opciones);
$smarty->assign('idSeleccionado', 4*10);

```

Y en el contenido del archivo que incluye al template:

```
{html_options options=$opciones selected=$idSeleccionado}
```

Las funciones **html_select_date** y **html_select_time** generan **dropdowns** de fecha y hora. Podemos consultar el manual de Smarty para obtener más información acerca de esto.

Para generar elementos **radio**, contamos con **html_radios**. Tiene los mismos atributos que **html_options** más **separator**:

```
{html_radios name="nombreRadio" options=$opciones selected=$idSeleccionado
separator=<br />"}
```

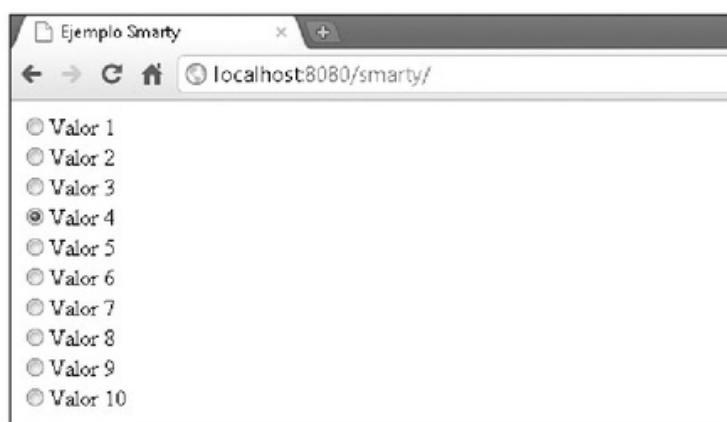


Figura 18. Con Smarty, es posible obtener datos desde un script PHP y asignarlos a elementos HTML.

```

<label><input type="radio" name="nombreRadio" value="10" />Valor
1</label><br />
<label><input type="radio" name="nombreRadio" value="20" />Valor
2</label><br />

```

```

<label><input type="radio" name="nombreRadio" value="30" />Valor
    3</label><br />
<label><input type="radio" name="nombreRadio" value="40" checked="checked" />
    Valor 4</label><br />
<label><input type="radio" name="nombreRadio" value="50" />Valor
    5</label><br />
<label><input type="radio" name="nombreRadio" value="60" />Valor
    6</label><br />
<label><input type="radio" name="nombreRadio" value="70" />Valor
    7</label><br />
<label><input type="radio" name="nombreRadio" value="80" />Valor
    8</label><br />
<label><input type="radio" name="nombreRadio" value="90" />Valor
    9</label><br />
<label><input type="radio" name="nombreRadio" value="100" />Valor
    10</label><br />

```

Para generar elementos **table**, contamos con **html_table**:

```

<?php

require 'smarty/lib/Smarty.class.php';
$smarty = new Smarty;

$smarty->template_dir = './smarty/templates';

$smarty->compile_dir = './smarty/templates_c';

$smarty->config_dir = './smarty/configs';
$smarty->cache_dir = './smarty/cache';

for ($c=1;$c<=10; $c++) {
    $datos[] = "Valor $c";
}

$smarty->assign('datos', $datos);

$smarty->display('index.tpl');

```

```
?>
```

- **index.tpl**

```
{html_table loop=$datos cols=6 table_attr='border="1" cellspacing="0"
 cellpadding="4"}
```

Salida:

```
<table border="1" cellspacing="0" cellpadding="4">
<tbody>
<tr>
<td>Valor 1</td>
<td>Valor 2</td>
<td>Valor 3</td>
<td>Valor 4</td>
<td>Valor 5</td>
<td>Valor 6</td>
</tr>
<tr>
<td>Valor 7</td>
<td>Valor 8</td>
<td>Valor 9</td>
<td>Valor 10</td>
<td>&nbsp;</td>
<td>&nbsp;</td>
</tr>
</tbody>
</table>
```

III FILTROS DE SMARTY

Debemos saber que Smarty permite incluir los llamados **prefiltros**, que son funciones PHP ejecutadas antes de compilar los templates. En el mismo sentido, están disponibles los **postfiltros**, ejecutados después de compilar los templates. Se trata de características avanzadas de la herramienta, que suelen utilizarse en proyectos de importancia.

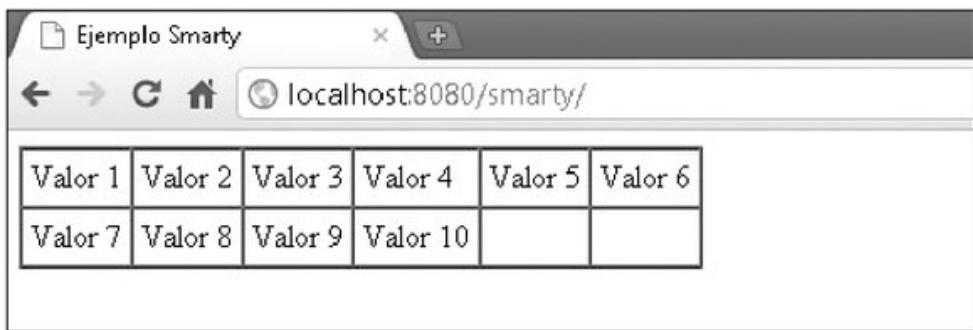


Figura 19. Las tablas admiten una gran cantidad de atributos, lo que se reduce en un alto control sobre la salida HTML.

Algunos de los atributos soportados por las tablas son **cols** (número de columnas), **rows** (número de filas), **loop** (fuente de datos), **table_attr** (atributos al elemento **table**), **tr_attr** (atributos al elemento **tr**), **td_attr** (atributos al elemento **td**) y **trail-pad** (valor de relleno para celdas vacías).

Caché

La caché es utilizada principalmente para alcanzar mayor velocidad en lo referido a la carga y visualización de templates, a través del método **display**, por ejemplo.

La definición del tiempo en caché de una página es importante y deberá ser personal para cada una. Así, un template que no se actualiza de manera frecuente podrá permanecer más tiempo en caché, y viceversa.

```
$smarty->caching = true;
```

Si la página no está en caché, y esta funcionalidad está activada, Smarty genera el archivo correspondiente y lo almacena en el directorio definido por la propiedad **cache_dir**. Los archivos generados son administrados por Smarty, y no hace falta realizarles ninguna modificación.

Podemos asignar a la propiedad **caching** el valor 2, lo que indica que el uso de la caché podrá ser personalizado para cada archivo de templates:

III ECUACIONES MATEMÁTICAS

La función denominada **math** nos permite generar ecuaciones matemáticas e incluirlas dentro del código fuente de un template. Esta función ofrece una gran cantidad de opciones, y su atributo más importante es **equation**, que se encarga de recibir como valor la siguiente ecuación: **{math equation="a / b + c" a=\$a b=\$b c=\$c}**.

```
$smarty->caching = 2;

$smarty->cache_lifetime = 120; //segundos
$smarty->display('index.tpl');

$smarty->cache_lifetime = 3600;
$smarty->display('noticias.tpl');

$smarty->cache_lifetime = 30;
$smarty->display('mensajes.tpl');
```

Si **display** recibe un segundo argumento, Smarty generará una caché distinta por cada valor. Los métodos **is_cached** y **clear_cache** también pueden recibir este parámetro extra:

```
$smarty->display('galeria.tpl', $idCliente);
```

Objetos

Otra funcionalidad con la que contamos es la de poder acceder a objetos PHP desde un template, previo registro de éstos. Veamos un ejemplo:

```
<?php

class nombreClase {
    function metodo1($arg1, $arg2) {
        return "Metodo #1 - ".($arg1 + $arg2);
    }

    function metodo2($arg1) {
        return "Metodo #2";
    }

    function metodo3($arg1) {
        return "Metodo #3";
    }
}
```

```
$nombreObjeto = new nombreClase;

$smarty->register_object("nombreObjetoEnTemplate", $nombreObjeto);

$smarty->display("index.tpl");

?>
```

- index.tpl

```
{nombreObjetoEnTemplate->metodo1 p1="2" p2="6" assign="salida"}
<b>{$salida}</b>
```

Ejemplo completo de uso

En el siguiente proyecto, implementaremos dos puntos: el ingreso de usuarios a través de un registro y la visualización del sitio en distintos idiomas. Utilizaremos:



Figura 20. Smarty nos induce a la programación en capas
y a la modularización de funciones.

- **include_base.tpl** (estructura que contiene a los demás templates).
- **include_encabezado.tpl** (encabezado de las páginas).
- **include_izquierda.tpl** (barra izquierda).

- **include_contenido.tpl** (contenido central).
- **include_pie.tpl** (pie de página).

La estructura de la base de datos para validar el nombre de usuario y la contraseña del usuario es la siguiente:

```
create database smarty;
use smarty;
create table users (
    idUser int(5) primary key not null auto_increment,
    nameUser varchar(200) not null,
    usernameUser varchar(200) not null,
    passwordUser varchar(200) not null
);

insert into users values (null, 'Fred Williams', 'fred@williams.com',
    'sdf5g2');
insert into users values (null, 'Michael Stephan',
    'mstephan@stepco.com', 'ad4g77');
```

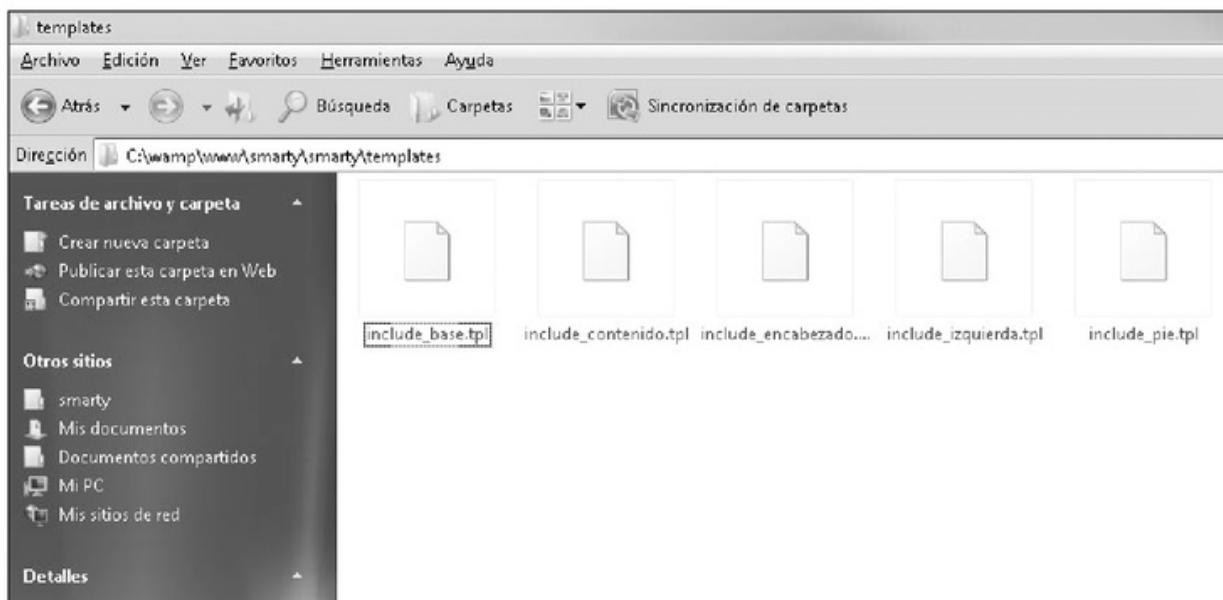


Figura 21. A través de Smarty, es posible estructurar sitios de manera que luego nos resulte sencillo mantenerlos.

El formulario de ingreso está en **include_izquierda.tpl**:

```
<form method="post">
```

```
<table width="160" border="0" cellspacing="0" cellpadding="2"
bordercolor="#FFFFFF" align="center" class="text">

{if $smarty.session.logged != ''}
<tr>
<td valign="top" bordercolor="#CCCCCC"> <a href="?logout=1">Logout</a>
</td>
</tr>
{else}
<tr>
<td valign="top" bordercolor="#CCCCCC"> Username<br>
<input type="text" name="usernameUser" name="usernameUser" value="$smarty.post.usernameUser" size="15">
<br>

</td>
</tr>
<tr>
<td valign="top" bordercolor="#CCCCCC">Password<br>
<input type="password" name="passwordUser" name="passwordUser" value="$smarty.post.passwordUser" size="15">
</td>
</tr>
<tr>
<td valign="top" bordercolor="#CCCCCC" height="33">
<input type="submit" id="sendLogin" name="sendLogin" value="Login">
</td>
</tr>

{if $smarty.post.sendLogin != ''}
<tr>
<td valign="top" bordercolor="#CCCCCC">
<p class="error">{$errorLogin}</p>
</td>
</tr>
{/if}
{/if}

</table>
</form>
```

Cuando se envía, el resultado se procesa en **index.php**:

```

if ($_POST['sendLogin']) {
    $res = mysql_query("select * from users where usernameUser = '$_POST
[usernameUser]' and passwordUser = '$_POST[passwordUser]'");
    if (mysql_num_rows($res)) {
        $row = mysql_fetch_array($res);
        $_SESSION['logged'] = true;
        $_SESSION['nameUser'] = $row['nameUser'];
        $_SESSION['dateLogged'] = date("d.m.Y h:iA");
        header("location: ?");
        exit;
    } else {
        $errorLogin = true;
    }
}
...
if ($errorLogin)
    $smarty->assign('errorLogin', $smarty->get_config_vars('errorLogin'));

```

Si los datos son incorrectos, mostramos un mensaje de error, caso contrario:

```

if ($_SESSION['nameUser'])
    $smarty->assign('nameUser', $_SESSION['nameUser']
        ($_SESSION['dateLogged']));

```

En **include_contenido.tpl** se toma una dirección u otra según el valor de la variable **logged**:

```

{if $smarty.session.logged != ''}
    <p>Lorem ipsum dolor sit amet, consetetur ...</p>

{else}
    <p>{#mensajeLogin#}</p>
{/if}

```

Cuando el usuario desea terminar su sesión, se ejecuta el siguiente código dentro de **index.php**:

```
session_start();

if ($_GET['logout']) {
    $tmp = $_SESSION['idioma'];
    session_unset();
    session_destroy();
    session_start();

    $_SESSION['idioma'] = $tmp;
    header("location: ?");
    exit;
}
```

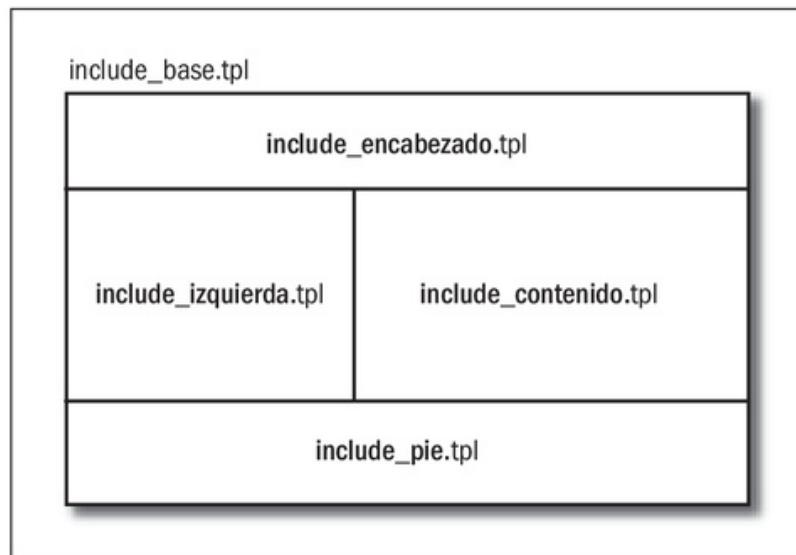


Figura 22. Smarty es una herramienta que nos permite enfrentar toda clase de proyectos, desde los simples hasta los complejos. Podemos importar indefinidas plantillas para cubrir nuestras necesidades.

Para cambiar de un idioma a otro, dentro del template `include_izquierda.tpl`, incluimos un combo con las opciones disponibles, que será el encargado de ofrecer al usuario los idiomas del sistema:

```
<select id="cboIdioma" name="cboIdioma" onchange="window.location =
'?cboIdioma=' + this.value">
```

```
{html_options options=$cboOpcionesIdioma selected=$cboIdIdiomaSeleccionado}
</select>
```

En **index.php**, almacenamos la selección en una variable de sesión:

```
if ($_GET['cboIdioma']) {
    $_SESSION['idioma'] = $_GET['cboIdioma'];
    header("location: ?");
    exit;
} elseif (!$SESSION['idioma']) {
    $_SESSION['idioma'] = 'esp';
}
```

Y cargamos los valores de las variables desde un archivo de configuración:

```
$smarty->config_load('idiomas.conf', $_SESSION[idioma]);

$cboOpcionesIdioma['eng'] = 'English / Ingles';

$cboOpcionesIdioma['esp'] = 'Spanish / Español';

$smarty->assign('cboOpcionesIdioma', $cboOpcionesIdioma);
$smarty->assign('cboIdIdiomaSeleccionado', $_SESSION['idioma']);
```

El archivo **idiomas.conf** tiene el siguiente aspecto:

```
[eng]
tituloSeleccioneIdioma = "Select your language: "
tituloInicioSesion = "Member Login: "
tituloPagina = "Smarty example"
tituloPaginaContenido = "Welcome to Company Website"
mensajeLogin = "Please insert your username and password"
errorLogin = "Login error !"
menuHome = "Home"
menuCompany = "Company"
menuServices = "Services"
menuProducts = "Products"
```

```

menuClients = "Clients"

menuDownloads = "Downloads"
menuSiteMap = "Site Map"
menuContactUs = "Contact Us"

[esp]
tituloSeleccioneIdioma = "Seleccione su idioma: "
tituloInicioSesion = "Inicio de Sesión: "
tituloPagina = "Ejemplo Smarty"
tituloPaginaContenido = "Bienvenido al Website de la Compañía"
mensajeLogin = "Por favor ingrese su nombre de usuario y contraseña"
errorLogin = "Error en registro !"
menuHome = "Inicio"
menuCompany = "Compañía"
menuServices = "Servicios"
menuProducts = "Productos"
menuClients = "Clientes"
menuDownloads = "Descargas"
menuSiteMap = "Mapa del Sitio"
menuContactUs = "Contacto"

```

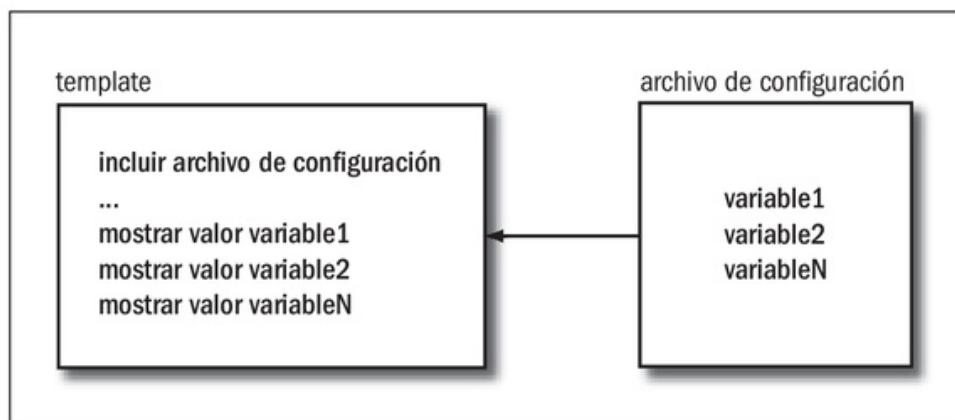


Figura 23. Podemos incluir y acceder a los archivos de configuración tanto desde un template como desde el código fuente de una aplicación.

Las variables están repartidas en los distintos templates. En cuanto a los archivos y directorios utilizados por el sistema, los definimos en **index.php** como vemos en el código a continuación:

```
include 'db/conexion.php';
```

```
require 'smarty/lib/Smarty.class.php';

$smarty = new Smarty;

$smarty->template_dir = './smarty/templates';
$smarty->compile_dir = './smarty/templates_c';
$smarty->config_dir = './smarty/configs';
```

La fuente del template que utilizamos en el proyecto es: www.templateyes.com.

Éste es sólo el comienzo. Smarty nos provee múltiples funcionalidades para cubrir las más diversas necesidades. Para obtener una referencia avanzada acerca de los métodos, propiedades, funciones y demás alternativas, podemos recurrir al manual de la aplicación en www.smarty.net/docs.php.



RESUMEN

Sin lugar a dudas, la utilización de templates da un marco profesional al desarrollo de aplicaciones web. Su puesta en práctica no genera mayores complicaciones y facilita el mantenimiento de un sitio. En este capítulo, se abarcaron sólo dos opciones de las muchas disponibles hoy en el mercado: Sigma, un proyecto generado y mantenido por PEAR, que permite un acercamiento amigable a la inclusión de templates en aplicaciones web, y Smarty, una opción avanzada de gran potencial.



ACTIVIDADES

PREGUNTAS TEÓRICAS

1 ¿Cuáles son las funciones básicas de un motor de templates?

2 ¿Qué diferencias encuentra entre Sigma y Smarty?

3 ¿Qué importancia le da a la caché de templates?

4 ¿En qué casos utilizaría templates?

5 ¿En qué casos no utilizaría templates?

6 ¿Cree que es acertada la inclusión de funciones de formato en un template?

7 ¿Para qué utiliza Smarty los archivos de configuración?

8 ¿Cuál cree que serían las ventajas de la utilización de templates para los programadores?

9 ¿Cuál cree que serían las ventajas de la utilización de templates para los diseñadores?

10 ¿Recomendaría el uso de templates?

EJERCICIOS PRÁCTICOS

1 Trate de implementar su propio motor de templates.

2 Desarrolle una misma aplicación con templates y sin ellos.

3 Actualice la aplicación anterior y verifique las ventajas y desventajas de la utilización de templates.

4 Calcule el rendimiento alcanzado utilizando caché para los templates.

5 Revise el contenido del directorio cache_dir de Smarty en sus aplicaciones.

Web 2.0: Ajax

Cada vez son más las aplicaciones que incluyen funcionalidades basadas en el modelo propuesto por Ajax, y PHP no podía estar al margen de esta nueva forma de interacción entre usuarios y sitios web. En este capítulo, recorremos algunas de las múltiples alternativas existentes para incluir Ajax en nuestros sistemas.

Origen y características	94
XML	100
JSON	102
Implementaciones en PHP	105
Sajax	106
Xajax	112
jPOP	122
Resumen	131
Actividades	132

ORIGEN Y CARACTERÍSTICAS

El término **Ajax** fue incluido por Jesse James Garrett (autor de diversos artículos técnicos, fundador del sitio www.adaptivepath.com) y es un acrónimo de *Asynchronous JavaScript And XML*. Ajax forma parte del movimiento conocido como **Web 2.0** o *Participatory Web* (**Web Participativa**) surgido a mediados de 2004 y que ha alcanzado una gran popularidad en los últimos años. Mantiene como principio fundamental la inclusión de herramientas y tecnologías que permitan la participación activa por parte del usuario en lo referido a sus experiencias interactuando con sitios web.

Una de las características más notables de la propuesta de Ajax es la de reutilizar tecnologías existentes y conocidas por un amplio porcentaje de desarrolladores, lo que nos permite un acercamiento amigable al modelo: herramientas como **(X)HTML**, **CSS**, **JavaScript**, **DHTML**, **DOM**, **XML**, **JSON**, o incluso **XSLT** llevan un tiempo considerable en el mercado, y el nivel de dificultad de cada una no es, por lo menos en principio, excesivamente alto.

Los desarrolladores Ajax no dependen de una empresa que les brinde, a partir de nuevas versiones de programas o **frameworks**, opciones o controles para lograr desarrollos más competitivos: las tecnologías son conocidas y públicas, no pertenecen a un particular, y el nivel dependerá de las cualidades de los programadores y diseñadores de aplicaciones. Pero más allá del aspecto técnico, Ajax pone énfasis en cómo se da la interacción entre un usuario y una aplicación web.

Por lo general, el cliente envía una petición al servidor, que luego del procesamiento correspondiente devuelve el resultado. Esto se mantiene en Ajax, pero la manera de recuperar la respuesta y de actualizar la información en la vista cambia. En el modelo tradicional se actualiza la página completa para ver reflejados los cambios. Esto supone que las partes que no se modifiquen, o sea, lo que no cambia entre la página desde la cual se realiza la petición y la que contiene la respuesta, también, serán recargadas. Se produce así una utilización mayor del ancho de banda disponible y una espera de respuesta excesiva para visualizar sólo unos pocos cambios.



III ORIGEN DE LA WEB 2.0

Es interesante saber que el término Web 2.0 fue incorporado por **O'Reilly Media** y **MediaLive International** en una serie de conferencias dictadas durante el año 2004. La comunidad lo adoptó rápidamente, y su mención representa una serie de ideas acerca de las características referidas a la interoperabilidad con un sitio determinado.

The screenshot shows a web page from adaptive path's 'ideas' section. At the top, there's a navigation bar with links for blog, about us, services, events, ideas, contact, and home. Below that is a secondary navigation bar with links for blog, book, podcasts, reports, essays and newsletter (which is highlighted in yellow), and reading list. The main content area features a black and white portrait of Jesse James Garrett, followed by his name and the date February 18, 2005. The article title is 'Ajax: A New Approach to Web Applications'. The text discusses how Ajax allows for rich, responsive web applications that can compete with desktop software. It mentions Google Suggest and Google Maps as examples. On the right side, there's a sidebar titled 'Recent Essays' with a list of articles and their dates.

Recent Essays

- A Blueprint for a Creative Workplace
July 15, 2010
- All y'all. Arguments and principles for doing research and design for families, couples and other independent groups.
June 2, 2010
- De koffie staat klaar! (coffee is ready!)
April 21, 2010
- Making Design Principles Stick
December 1, 2009
- An Interview about Sonic Branding with Martyn Ware, Founder of SonicID
November 17, 2009
- [Essay Archives >>](#)
- [Newsletter Archives >>](#)

Figura 1. Ajax establece la forma de conexión entre un usuario y una aplicación web.

En Ajax, la parte que más consumo de recursos demanda es la carga inicial de la página, o sea la interfaz de usuario. Una vez que esto sucede, la aplicación realiza peticiones al servidor y actualiza la interfaz con los datos recuperados. Las vistas no están compuestas sólo por la presentación (html, hojas de estilo, etcétera), sino también por las funciones JavaScript para actualizar los datos contenidos en ellas. En resumen, la interfaz debe:

- Brindar opciones para realizar peticiones.
- Recuperar las respuestas.
- Tratarlas y aplicarles formato.
- Actualizar la vista del usuario.

La implementación de soluciones Ajax no está ligada a un lenguaje de programación en particular, es decir que no contamos con una única manera de llevar a cabo una tarea: todo queda en manos del programador quien tendrá la libertad de orientar la acción hacia un punto u otro.

La clave de Ajax es poder acceder a partes de la presentación, definida quizá mediante **(X)HTML** (*eXtensible HyperText Markup Language*) y **CSS** (*Cascading Style Sheets*),

modificando el contenido de elementos mediante **DHTML** (*Dynamic HTML*). Para localizarlos existe **DOM** (*Document Object Model*), disponible desde JavaScript, entre otros lenguajes. Los datos recuperados desde el servidor pueden estar en formato **XML** (*eXtensible Markup Language*) o **JSON** (*JavaScript Object Notation*), por ejemplo, y ser tratados con **XSLT** (*eXtensible Stylesheet Language for Transformations*) o simplemente funciones JavaScript en el lado cliente. Ajax propone una idea, luego las tecnologías que se utilicen para implementarla dependerán de cada caso en particular. Esto se realizaba tradicionalmente en el lado cliente, y Ajax introdujo la alternativa de recuperar información desde el lado servidor, y actualizar la página de manera **asincrónica**, es decir, sólo partes de ella: JavaScript permite a través de un objeto llamado **XMLHttpRequest** enviar y recibir datos entre un navegador y un servidor web, creando un canal de conexión independiente (asincrónico) entre el cliente y el servidor.

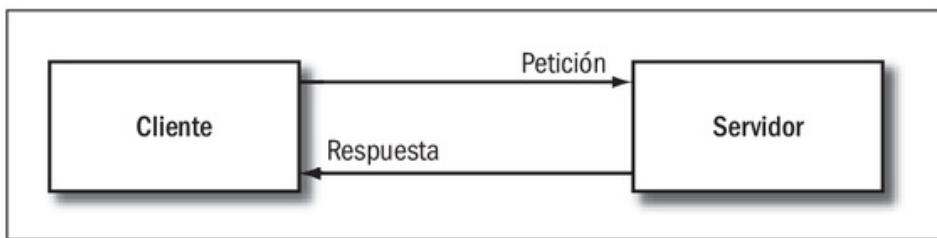


Figura 2. Ajax mantiene la estructura cliente-servidor tradicional;
lo que cambia es la forma de establecer la comunicación.

- Con **sincronía**, se expresa la dependencia entre procesos (cuando un navegador realiza una petición, la actividad del usuario se interrumpe hasta que se recupere la respuesta).
- La **asincronía** puede definirse como la capacidad que una aplicación tiene de administrar procesos independientes unos de otros (la actividad del usuario no se interrumpe totalmente debido a que una de las cualidades de Ajax consiste en mantener la interfaz de usuario siempre visible y bloquear en forma transitoria sólo una parte de ella, la que se actualizará al momento de recuperar la respuesta desde el servidor, algo similar a lo que sucede con las aplicaciones de escritorio). Un cliente recurre al servidor sólo cuando es absolutamente necesario hacerlo.



DEFINICIÓN DE AJAX

Podemos observar a Ajax como un claro ejemplo de arquitectura de sitios web o también como un conjunto de herramientas específicas para implementar dicha arquitectura. Si buscamos más información al respecto, podemos encontrarla en el artículo disponible en www.adaptivepath.com, allí se incluyen las dos alternativas para definirlo.

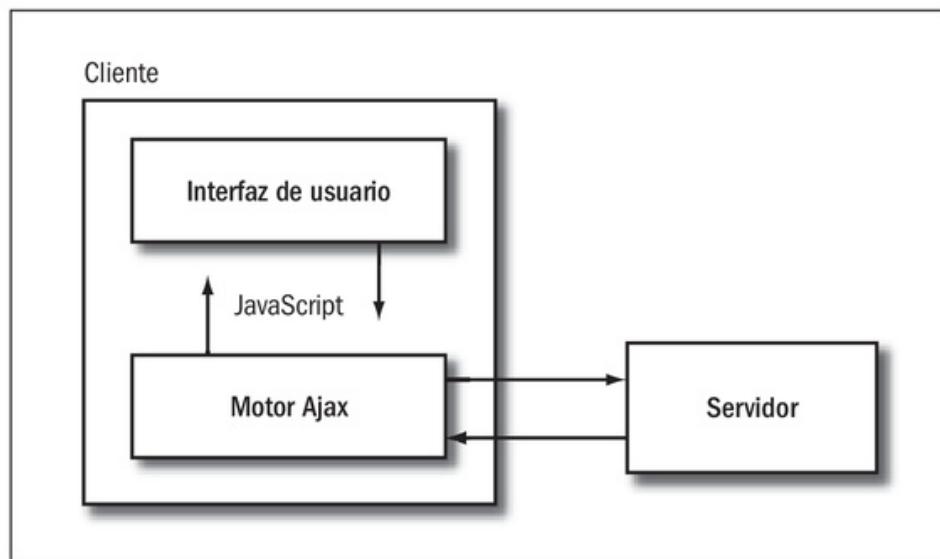


Figura 3. La parte cliente juega en Ajax un papel fundamental puesto que, desde allí, se deben realizar las peticiones y las actualizaciones en la presentación.

En una aplicación Ajax, luego de crear una instancia del objeto **XMLHttpRequest** (algo así como un intermediario entre la interfaz de usuario y el servidor), se prepara la petición, se la envía al servidor y se recibe una respuesta, que luego servirá para actualizar (a través del DOM de JavaScript) la interfaz del cliente. En el servidor, podemos recuperar la información desde una base de datos, un servicio web, archivos planos, o generarla a partir de lenguajes como PHP, JSP, o ASP .NET, por ejemplo. La salida será retornada luego a la aplicación cliente desde la cual se generó la petición.

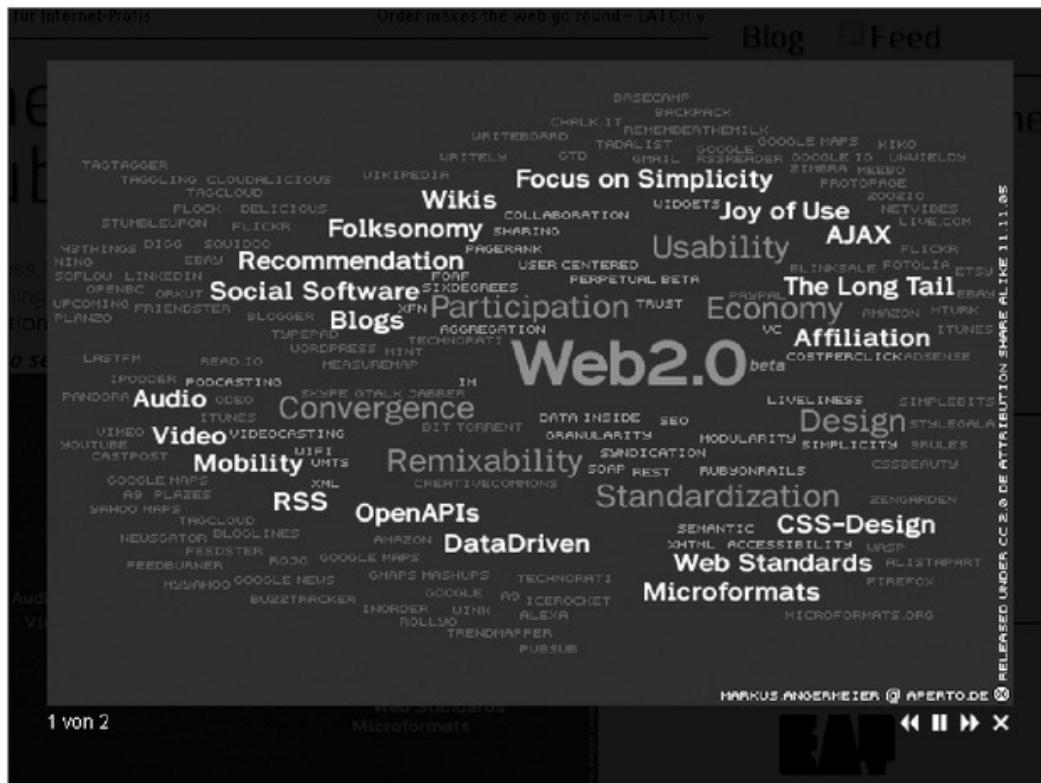


Figura 4. El movimiento conocido como Web 2.0 adquirió gran popularidad en los últimos años.

El objeto no es reciente, pero su momento de popularidad llegó al convertirse en una pieza clave del modelo propuesto por Ajax. Está disponible no sólo en JavaScript, sino en otros lenguajes como **JScript** y **VBScript**. Aun cuando la **W3C** está trabajando para establecer un estándar para su implementación, por el momento, cada navegador posee la suya propia. **Microsoft** fue la primera empresa en introducirlo en sus navegadores mediante un objeto **ActiveX**. **Internet Explorer** da soporte desde la versión 5 (objeto **XMLHTTP**), y a partir de la versión 7 incluye **XMLHttpRequest** como un objeto JavaScript nativo. La funcionalidad del objeto nativo y el ActiveX es similar (no idéntica), y las mayores diferencias se dan en la velocidad de ejecución alcanzada. Otros navegadores que dan soporte son los basados en el motor **Gecko**, como por ejemplo, **Mozilla Firefox**, **Camino**, **Galeon**, **Netscape** 7.1 y superiores, **Konqueror** versión 3.2 y superiores, **Safari** de Apple versión 1.2 y superiores, y **Opera** versión 8.0 y superiores.

Aunque estandarizar y documentar las funcionalidades, características y formas de acceso (propiedades, métodos y eventos) de este objeto nos permitirá un trabajo más sencillo, actualmente las librerías/frameworks existentes encapsulan los mecanismos de creación y tratamiento, para evitarnos la necesidad de generar código que detecte el navegador que utiliza el usuario y de escribir código específico para cada caso.

W3C Candidate Recommendation

XMLHttpRequest

W3C Candidate Recommendation 3 August 2010

This Version:
<http://www.w3.org/TR/2010/CR-XMLHttpRequest-20100803/>

Latest Version:
<http://www.w3.org/TR/XMLHttpRequest/>

Latest Editor Version:
<http://dev.w3.org/2006/webapi/XMLHttpRequest/>

Previous Versions:

- <http://www.w3.org/TR/2009/WD-XMLHttpRequest-20091119/>
- <http://www.w3.org/TR/2009/WD-XMLHttpRequest-20090820/>
- <http://www.w3.org/TR/2008/WD-XMLHttpRequest-20080415/>
- <http://www.w3.org/TR/2007/WD-XMLHttpRequest-20071026/>
- <http://www.w3.org/TR/2007/WD-XMLHttpRequest-20070618/>
- <http://www.w3.org/TR/2007/WD-XMLHttpRequest-20070227/>
- <http://www.w3.org/TR/2006/WD-XMLHttpRequest-20060927/>
- <http://www.w3.org/TR/2006/WD-XMLHttpRequest-20060619/>
- <http://www.w3.org/TR/2006/WD-XMLHttpRequest-20060405/>

Editor:
Anne van Kesteren (Opera Software ASA) <annevk@opera.com>

Copyright © 2009 W3C® (MIT, ERCIM, Keio). All Rights Reserved. W3C liability, trademark and document use rules apply.

Figura 5. Internet Explorer fue uno de los primeros navegadores en implementar el objeto XMLHttpRequest. Hoy en día, la mayoría lo hace.

```

<script type="text/javascript">
<!--

var XMLHttpRequest=null;

if (window.XMLHttpRequest) {
    XMLHttpRequest = new XMLHttpRequest()
    alert('Mozilla / Safari / IE7 / etc')
} else if (window.ActiveXObject) {
    XMLHttpRequest = new ActiveXObject("Microsoft.XMLHTTP")
    alert('Internet Explorer')
}

if (!XMLHttpRequest) {
    alert('No se pudo crear la instancia');
}

//-->
</script>

```

Prototype (www.prototypejs.org), uno de los frameworks JavaScript que implementa Ajax, inicializa un objeto **XMLHttpRequest** de la siguiente manera:

```

var Ajax = {
    getTransport: function() {
        return Try.these(
            function() {return new XMLHttpRequest()},
            function() {return new ActiveXObject('Msxml2.XMLHTTP')},
            function() {return new ActiveXObject('Microsoft.XMLHTTP')}
        ) || false;
    }
}

```

III EL API DE XMLHTTPREQUEST

El *World Wide Web Consortium* ha publicado un documento (en el año 2006) que en la actualidad está en revisión, completado y modificado para especificar el API (*Application Programming Interface*) del objeto **XMLHttpRequest**. Una vez finalizado este proceso, las empresas que desarrollan los navegadores web podrán guiarse a partir de los resultados obtenidos.

```
 },
activeRequestCount: 0
};
```

Como vemos, para que las aplicaciones basadas en Ajax funcionen correctamente, el cliente deberá acceder a ellas a través de un navegador con soporte JavaScript habilitado y que admita la posibilidad de generar objetos **XMLHttpRequest**.

El formato del texto retornado como respuesta podrá ser de cualquier tipo, por ejemplo, texto plano, XML o JSON (veremos en detalle estas alternativas más adelante en este mismo capítulo). Aun cuando la respuesta pueda contener información de cualquier tipo, la idea propuesta por Ajax es que el contenido y la presentación estén separados, es decir, que el formato de la respuesta debería ser definido en el lado cliente, a través de CSS, por ejemplo. Además, las respuestas recuperadas desde el servidor tienen que ser preferentemente pequeñas en tamaño (por lo general son en formato de texto plano) para que la actualización no demore un tiempo excesivo.

XML

XML (*eXtensive Markup Language*, Lenguaje de Marcado eXtensible) es un metalenguaje cuyo objetivo consiste en posibilitar el intercambio de información entre aplicaciones o almacenar datos. A pesar de que forma parte del acrónimo de Ajax, no es la única opción para intercambiar datos con el servidor.

Para que un documento XML se considere bien formado, deberá respetar algunas reglas sintácticas que permitirán luego que un parseador XML, o sea una aplicación que intenta leer un documento en este formato, pueda recorrerlo e interpretar su contenido sin problemas.

III COMPARACIÓN

Debemos mencionar que Ajax intenta brindar la sensación ofrecida por una aplicación de escritorio sin perder las posibilidades alcanzadas hoy en día por las difundidas aplicaciones web. En una aplicación de escritorio, el usuario no pierde contacto con la interfaz, aun cuando ésta se actualiza en línea o se encarga de procesar información.

La estructura de un documento XML se basa en etiquetas. Este metalenguaje ha sido tomado como una opción válida por sobre HTML principalmente porque mientras este último define el formato de presentación, XML permite especificar la estructura de los datos contenidos, de alguna manera describir su significado. Además, en XML, no hay etiquetas predefinidas. Los elementos son creados por el autor del documento quien estructura la información según sus necesidades. Sin embargo, XML no fue desarrollado con la idea de reemplazar a HTML, sus propósitos iniciales son totalmente diferentes. XHTML sí podría emparentársele.

Estás en: Oficina Española > Documentos y Guías > Guías Breves > XML

W3C WORLD WIDE WEB
consortium
Oficina Española

[Volver al índice](#)

Guía Breve de Tecnologías XML

[[¿Qué es?](#) | [Q](#)] [[¿Para qué sirve?](#) | [P](#)] [[¿Cómo funciona?](#) | [C](#)] [[Ejemplos](#) | [E](#)] [[Más información](#) | [M](#)]

¿Qué son las Tecnologías XML?

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de datos. Es un lenguaje muy similar a [HTML](#), pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones.

Las tecnologías XML son un conjunto de módulos que ofrecen servicios útiles a las demandas más frecuentes por parte de los usuarios. XML sirve para estructurar, almacenar e intercambiar información.

¿Para qué sirven?

Entre las tecnologías XML disponibles se pueden destacar:

XSL: Lenguaje Extensible de Hojas de Estilo, cuyo objetivo principal es mostrar cómo debería estar estructurado el contenido, cómo debería ser diseñado el contenido de origen y cómo debería ser paginado en un medio de presentación como puede ser una ventana de un navegador Web o un dispositivo móvil, o un conjunto de páginas de un catálogo, informe o libro.

XPath: Lenguaje de Rutas XML, es un lenguaje para acceder a partes de un documento XML.

Figura 6. XML fue uno de los primeros formatos masivos para intercambio de datos entre aplicaciones.

Las reglas anteriormente mencionadas son intuitivas y fáciles de incorporar en el desarrollo diario, por ejemplo:

- Declarar al inicio del archivo que se trata de un documento con contenido XML (de la forma `<?xml version="1.0"?>`). Además de la versión de XML por utilizar, podemos incluir opcionalmente el juego de caracteres (`<?xml version="1.0" encoding="utf-8"?>`). Si no especificamos **encoding**, el parser XML asume que los caracteres pertenecen al conjunto UTF-8.
- Cada documento tendrá un elemento raíz que contenga a todos los demás. Un archivo XML debe incluir obligatoriamente uno o más elementos (un documento no puede estar vacío).

- Los nombres de los elementos son sensibles a mayúsculas.
- Incluir etiquetas de cierre, incluso si se trata de elementos vacíos. Los elementos se cierran en el orden inverso al que fueron abiertos.
- Los nombres de los elementos y atributos no pueden contener espacios.
- Los comentarios son como los de HTML.

JSON

Esta opción quizá sea más popular que XML. **JSON** (*JavaScript Object Notation*, Notación en Objetos de JavaScript) es una sintaxis (o notación) para representar información. No es un programa ni un lenguaje de programación particular, es sólo una manera de describir y estructurar datos que en forma eventual podrán ser intercambiados entre aplicaciones.

Su sintaxis es considerada completamente independiente de cualquier lenguaje de programación y consiste en pares nombre/valor y listas ordenadas de valores (arrays) como veremos a continuación.

```
{
  usuario: {
    nombre: 'Amy',
    apellido: 'Lee',
    telefono: [
      {
        numero: '444-4444',
        tipo: 'hogar'
      },
      {
        numero: '555-5555',
        tipo: 'oficina'
      }
    ]
  }
}
```



USOS DE JSON

Cada vez son más los servicios web que incorporan a JSON para realizar el intercambio de información. En este sentido **Yahoo** se presenta como una de las empresas pioneras en este aspecto (<http://developer.yahoo.com/yui/json>) y en sus desarrollos podemos observar cada vez más a menudo la implementación de este formato de intercambio y almacenamiento de datos.

```

        tipo:'trabajo'
    }
]
}
}

```

JSON, al igual que XML, necesita que las aplicaciones que participan en el intercambio puedan comprender e interpretar a la perfección los paquetes recibidos y generar otros nuevos para enviar.

La estructura propuesta por JSON es un objeto raíz que puede contener otros objetos, arrays, cadenas de caracteres, números, booleanos (verdadero / falso), o nulos. Un documento en este formato cuenta con algunos símbolos que delimitan partes con significado diferente, como vemos en el listado a continuación:

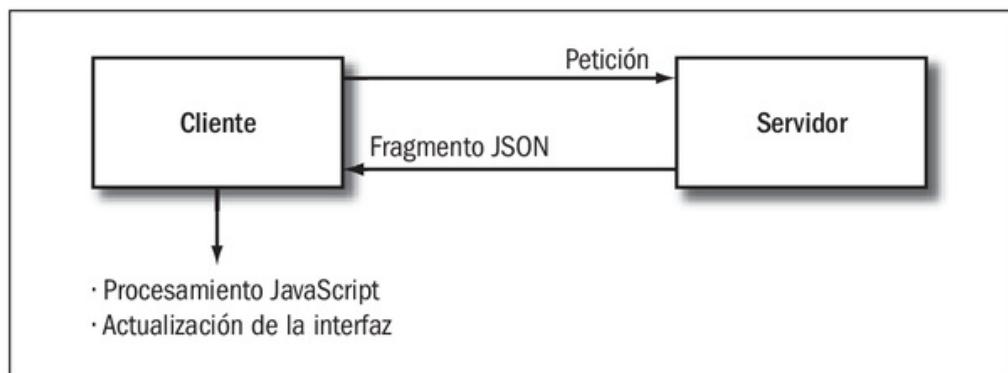


Figura 7. Las estructuras JSON pueden formar parte de la comunicación establecida entre un cliente y un servidor.

- Las llaves (`{}`) actúan como contenedores, definen las jerarquías que nos permitirán acceder a distintas partes de la estructura para recuperar un dato en particular.
- Los corchetes (`[]`) definen arrays, y usualmente los utilizaremos para representar elementos repetitivos.
- Los dos puntos (`:`) separan los pares nombre/valor.
- Las comas (`,`) separan los elementos de un array, en caso de haber más de uno, y los pares nombre / valor.
- En XML es posible utilizar cualquier nombre para identificar elementos (siguiendo ciertas reglas, como no incluir espacios o símbolos) mientras que en JSON no es posible utilizar palabras reservadas del lenguaje JavaScript.

JSON utiliza la notación literal de objetos de JavaScript, algo así como un subconjunto del lenguaje. Aunque no depende de él, el acceso a fragmentos estructurados en JSON desde un script JS es sencillo y nada problemático, contrariamente a lo que sucede con XML.

Forma tradicional:

```
var objeto = new Object();
objeto.valor = "xxx";
objeto.getValor = function() { return this.valor; }
```

Forma alternativa (utilizando la notación literal JSON):

```
var objeto = {
    valor: "xxx",
    getValor: function() { return this.valor; }
}
```

Podemos analizar una estructura JSON en JavaScript sólo utilizando la función **eval** o a través de una clase disponible en el sitio oficial de JSON, incluida en el archivo **json.js**, que permite convertir desde/hacia este formato.

The screenshot shows the homepage of json.org. On the left is a large black 'O' logo. To its right is a white rectangular box containing the title 'Introducing JSON'. Below the title is a horizontal bar with language links: Arabic, Bulgarian, Chinese, Czech, Dutch, English, Esperanto, French, German, Greek, Hebrew, Hungarian, Indonesian, Italian, Japanese, Korean, Persian, Polish, Portuguese, Russian, Slovenian, Spanish, Turkish, and Vietnamese. The main content area starts with a paragraph about JSON being a lightweight data-interchange format. It then lists two structures: objects (name/value pairs) and arrays (ordered lists). A sidebar on the right defines the JSON grammar terms: object ({}), members (pair or pair, members), pair (string : value), array ([]), elements (value or value, elements), and value (string, number, object, or array).

object
 {}
 members
 pair
 pair , members

pair
 string : value

array
 []
 elements
 value
 value , elements

value
 string
 number
 object
 array

Figura 8. JSON nos provee implementaciones para codificar/decodificar estructuras en una gran cantidad de lenguajes.

A su vez, hay implementaciones para otros lenguajes además de JavaScript, que podemos encontrar en www.json.org.

Implementaciones en PHP

Como consecuencia de la gran popularidad del lenguaje, no tardaron en surgir numerosos entornos de programación y librerías que nos permiten implementar soluciones Ajax en PHP. Algunas de las opciones disponibles son:

NOMBRE	SITIO WEB
AJASON	http://sourceforge.net/projects/ajason
Ajax Agent	http://ajaxagent.org
AjaxAC	http://ajax.zervaas.com.au
Cajax	http://sourceforge.net/projects/cajax
Claw	http://claw.tigris.org
DutchPIPE	www.dutchpipe.org
Flexible Ajax	http://sourceforge.net/projects/fixajax
HTML_AJAX (PEAR)	http://pear.php.net/package/HTML_AJAX
HTS	www.htsdesign.com
jPOP	http://jpop-framework.sourceforge.net
My-BIC	http://lifefuel.net/mybic
NanoAjax	www.nanoajax.org
PAJAX	www.auberger.com/pajax
phpAjaxTags	http://ajaxtags.sourceforge.net
PHPWebBuilder	http://groups.google.com/group/phpwebbuilder
Pipeline	http://livepipe.net
Qcodo	www.qcodo.com
Sajax	http://absinth.modernmethod.com/sajax
SimpleJax	http://simplejax.sourceforge.net
Symfony	www.symfony-project.com
TinyAjax	http://codeigniter.com/wiki/TinyAjax
XAJAX	http://www.xajax-project.org
Zephyr	http://zephyr-php.sourceforge.net

Tabla 1. Frameworks disponibles para PHP.

Dado que todos los días aparecen más y más frameworks de Ajax para ser utilizados con el lenguaje de programación PHP, es importante saber que en la tabla anterior vimos sólo un pequeño extracto de las alternativas que actualmente se ofrecen a los usuarios a través de Internet. En las siguientes páginas de este capítulo, nos dedicaremos a describir y analizar tres de las tantas opciones disponibles para el usuario; éstas son las siguientes: **Sajax**, **Xajax** y **jPOP**.

SAJAX

Sajax (*Simple Ajax*) es una herramienta que nos permite desarrollar aplicaciones web utilizando Ajax. Además de PHP, incluye soporte para **ASP**, **Cold Fusion**, **Io**, **Lua**, **Perl**, **Python** y **Ruby**. Es software libre y se distribuye bajo la licencia **BSD**. Podemos encontrar más información en <http://absinth.modernmethod.com/sajax>. Es compatible con los siguientes navegadores web: Internet Explorer, Mozilla Firefox, Safari y también Opera.

The screenshot shows the Sajax homepage with a navigation bar at the top. Below the navigation, there's a main content area with a heading "Overview of Ajax technology". To the right of this heading is a diagram illustrating the Ajax architecture. The diagram shows a "Database server" connected to an "Apache Webserver" which runs "App/CGI.PHP". An arrow points from the "Apache Webserver" to a "Web browser (IE, Firefox, Safari) Refreshless browsing". Below the browser is the text "JavaScript". Arrows also point from the "Database server" and the "Apache Webserver" to the "JavaScript" box. On the left side of the page, there's a "Welcome to Sajax!" message, some commercial consulting information, a news update about version 0.12, and sections for "EXAMPLES" and "DOWNLOAD NOW".

Figura 9. Sajax propone una implementación relativamente simple, libre de inconvenientes, para aquellos que quieran experimentar con Ajax sin poseer conocimientos profundos.

Para trabajar con Sajax, tendremos que definir funciones en PHP, que deberemos registrar previamente para ser aceptadas por la aplicación cliente.

Las funciones en el lado cliente tienen el mismo nombre que las funciones del lado servidor, pero precedidas por `x_` (x guión bajo). El último argumento a las funciones `x_` es el nombre de la función JavaScript que recogerá el resultado obtenido como respuesta (función `callback`). A continuación veremos una serie de ejemplos donde presentaremos tanto las funciones del lado del cliente, como las funciones del lado del servidor, y la interacción entre ellas a través de Sajax.

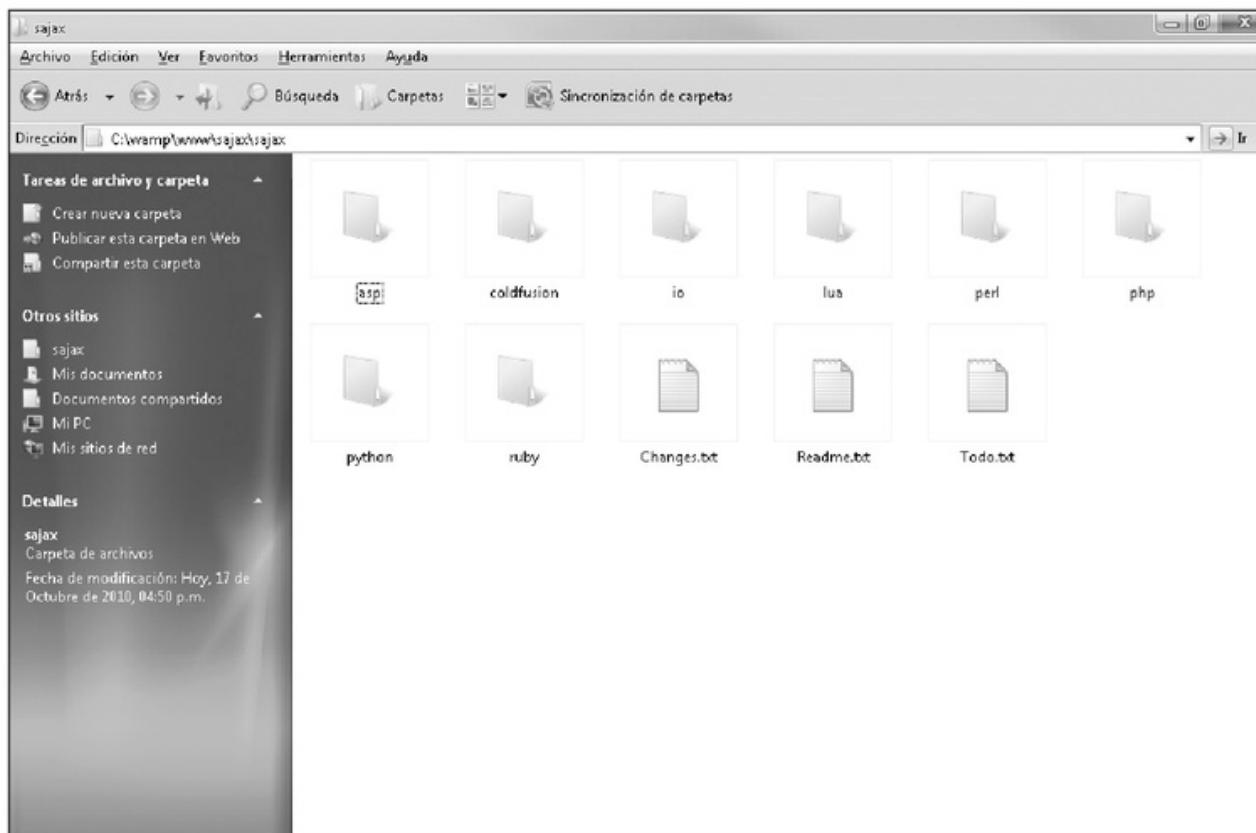


Figura 10. Sajax está disponible para múltiples lenguajes además de PHP. Todas las opciones se incluyen en la distribución oficial.

Sajax se encarga de crear el código necesario para que podamos acceder desde el lado cliente a las funciones PHP registradas. Veamos un ejemplo:

```
<?php

require_once("sajax/php/Sajax.php");

function obtenerFecha($nombre) {
    if (!$nombre) $nombre = 'Anonimo';
    return "Hola $nombre, es la hora ".date("h:i A");
```

III APPLICACIONES CON AJAX

Es importante saber que aplicaciones como **Gmail**, **Google Maps**, **Google Suggest** (Google es una de las empresas que más apoya el uso de Ajax, lo que demuestra en sus aplicaciones), o también **Flickr** (www.flickr.com), entre muchas otras, han sido desarrolladas utilizando Ajax. El uso de Ajax puede ser advertido a simple vista, sólo con utilizarlas.

```
}

sajax_init();
sajax_export("obtenerFecha");
sajax_handle_client_request();

?>
<html>
<head>
    <title> sajax </title>

    <script language="javascript">
        <?php sajax_show_javascript(); ?>

        function contenedorRespuesta(resultado) {
            document.getElementById("respuesta").innerHTML = resultado;
        }

        function generarPeticion() {
            var nombre = document.getElementById("nombre").value;
            x_obtenerFecha(nombre, contenedorRespuesta);
        }
    </script>

    <style>
        div {
            width: 400px;
            border: 1px solid #c0c0c0;
            text-align: center;
            padding: 14px;
            font-family: Arial, Helvetica, sans-serif;
            font-size: 11px;
            color: #000;
            margin-bottom: 4px;
        }

        #respuesta {
            font-weight: bold;
        }
    </style>
```

```

</head>
<body>

<div>
    ingrese su nombre :
    <input type="text" id="nombre">
    <input type="button" value="Enviar" onclick="generarPeticion();">
</div>

<div id="respuesta"></div>

</body>
</html>

```

Las aplicaciones que utilizan Sajax cumplen una serie de pasos predeterminados:

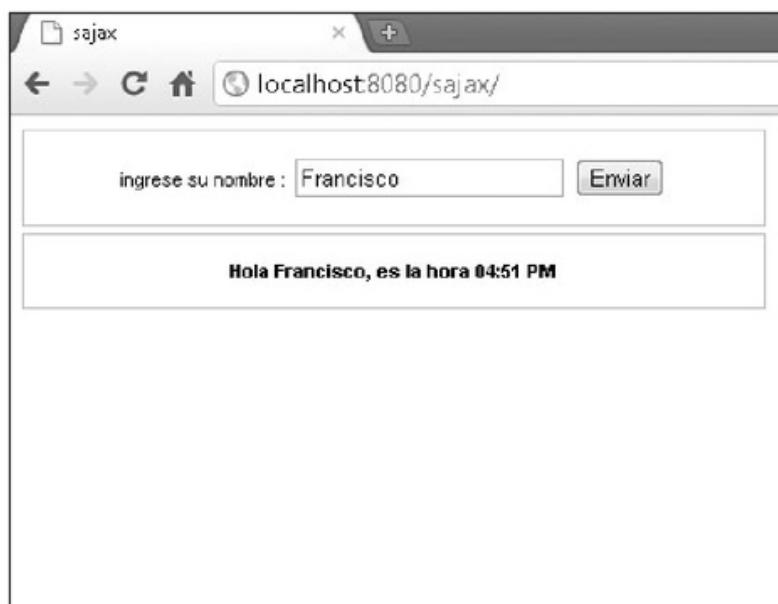


Figura 11. A través de Sajax, podemos invocar funciones PHP desde el lado cliente.

- 1) Incluir la librería **Sajax.php**:

```
require_once("sajax/php/Sajax.php");
```

- 2) Definir las funciones PHP que utilizaremos:

```
function obtenerFecha($nombre) {
```

```

if (!$nombre) $nombre = 'Anonimo';
return "Hola $nombre, es la hora ".date("h:i A");
}

```

3) Exportarlas:

```

sajax_init();
sajax_export("obtenerFecha");
sajax_handle_client_request();

```

4) Generar el código JavaScript necesario:

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

5) Anteponer el prefijo **x_** al momento de llamar a las funciones exportadas y agregar como último parámetro el nombre de la función JavaScript que va a recibir el resultado:

```
x_obtenerFecha(nombre, contenedorRespuesta);
```

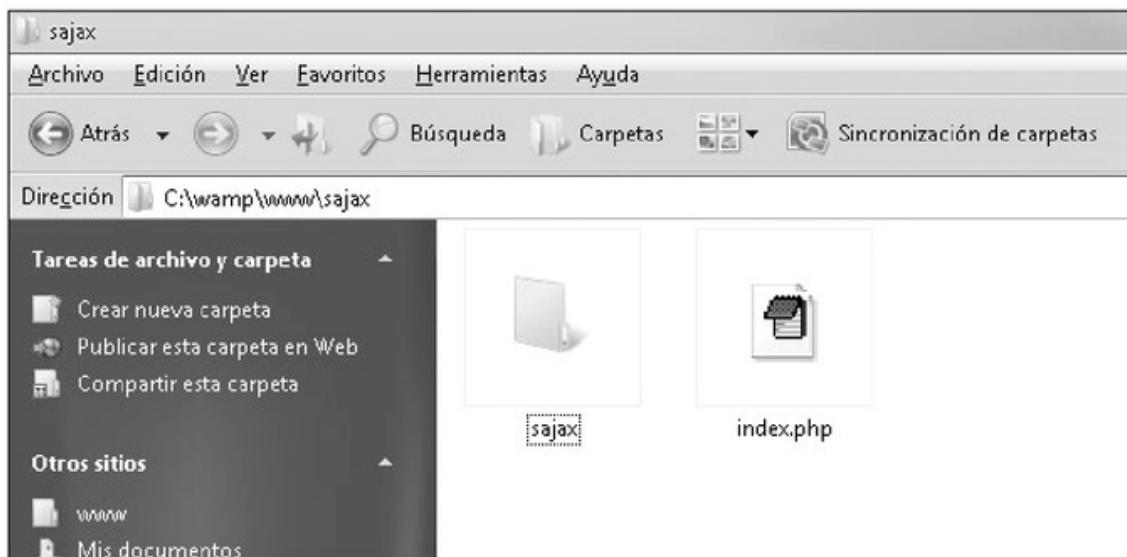


Figura 12. Para utilizar Sajax, debemos incluir la distribución en un directorio accesible por las páginas PHP del sitio.

La llamada a la función **contenedorRespuesta** es automática y se produce al recibir el resultado de la petición.

```
//Llamada Automática
function contenedorRespuesta(resultado) {
    document.getElementById("respuesta").innerHTML = resultado;
}
```

Si visualizamos el código fuente de la página a través de un explorador web, veremos cómo Sajax genera, entre otras cosas, una instancia del objeto **XMLHttpRequest** para interactuar con el lado servidor.



The screenshot shows a browser window with the title "sajax" and the URL "view-source:localhost:8080/sajax/". The content of the page is the raw HTML source code of the Sajax application. The code includes a title element with the text "sajax", a script element containing the Sajax library code, and a comment indicating the copyright of modernmethod, inc. The library code handles XMLHttpRequest creation, debugging, and initialization.

```

<html>
<head>
<title> sajax </title>
<script language="javascript">

    // remote scripting library
    // (c) copyright 2005 modernmethod, inc
    var sajax_debug_mode = false;
    var sajax_request_type = "GET";
    var sajax_target_id = "";
    var sajax_failure_redirect = "";

    function sajax_debug(text) {
        if (sajax_debug_mode)
            alert(text);
    }

    function sajax_init_object() {
        sajax_debug("sajax_init_object() called...")
        var A;

        var msxmlhttp = new Array(
            'Msxml2.XMLHTTP.5.0',
            'Msxml2.XMLHTTP.4.0',
            'Msxml2.XMLHTTP.3.0',
            'Msxml2.XMLHTTP',
            'Microsoft.XMLHTTP');
        for (var i = 0; i < msxmlhttp.length; i++) {
            try {
                A = new ActiveXObject(msxmlhttp[i]);
            } catch (e) {
                A = null;
            }
        }

        if (!A && typeof XMLHttpRequest != "undefined")
            A = new XMLHttpRequest();
        if (!A)
            sajax_debug("Could not create connection object.");
    }

```

Figura 13. Sajax utiliza la funcionalidad del objeto XMLHttpRequest para su funcionamiento.



EL USO DE XMLHttpRequest

Debemos saber que una aplicación Ajax se ejecuta en entornos web, más precisamente en navegadores que brinden un soporte completo para el objeto **XMLHttpRequest** de JavaScript. Si bien el uso de este objeto no se presenta como obligatorio, la mayoría de las aplicaciones lo utilizan para realizar la implementación de Ajax.

Para activar el modo debug en Sajax, debemos incluir el archivo **Sajax.php** y luego definir como verdadera (**1, true**) la variable **\$sajax_debug_mode**, que de manera pre-determinada está desactivada:

```
$sajax_debug_mode = 1;
```

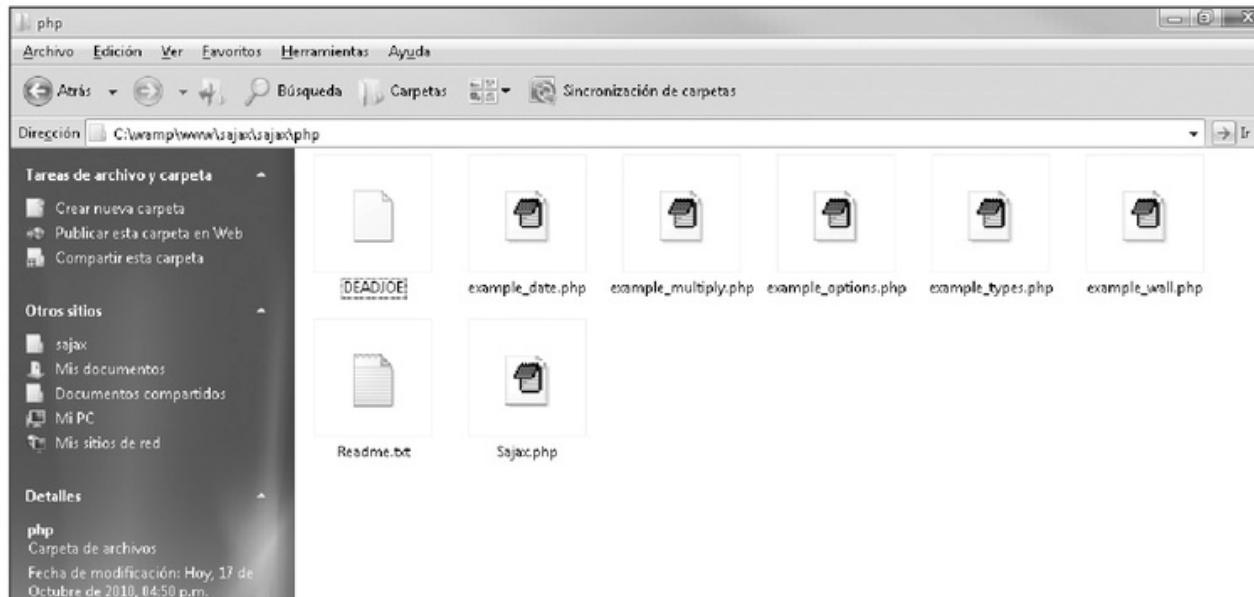


Figura 14. La distribución Sajax contiene ejemplos de uso completos listos para implementar.

En resumen, Sajax nos propone una manera simple y rápida de implementar Ajax en aplicaciones PHP.

XAJAX

A diferencia de Sajax, Xajax (www.xajax-project.org) es una librería orientada a objetos. Admite más opciones y conserva la facilidad para implementar Ajax.



VERSIÓN DE XAJAX

La librería denominada Xajax ha sufrido una serie de importantes cambios desde sus primeras versiones, por lo que actualizar desarrollos puede no resultarnos tan sencillo, dado que se han modificado los nombres de los métodos y sus respectivos argumentos. Para saber qué versión estamos utilizando, existe el método **getVersion**.



Figura 15. Xajax es una opción en pleno proceso de expansión. Podemos visitar periódicamente el sitio oficial en busca de las últimas noticias.

Xajax funciona con **Apache** o **IIS** (*Internet Information Server*, para Windows XP o superiores) y precisa de PHP versión 4.3 o superiores. En lo referido a los navegadores, soporta Internet Explorer 5.5 y superiores, Firefox 1.0 y equivalentes, Safari 1.3, y Opera 8.5. De todos modos, no debemos olvidarnos de que el soporte para ésta y las demás librerías se actualiza de manera permanente, por lo que recomendamos acceder a los sitios oficiales para obtener la información de última hora al respecto.

Para disponer de Xajax en una aplicación, primero debemos descargar la distribución, copiarla a un directorio accesible desde las demás páginas, e incluir el archivo **xajax.inc.php** en cada una de ellas:

```
require_once("xajax.inc.php");
```

Y luego crear una instancia de la clase **xajax**:

```
$xajax = new xajax();
```

Nuevamente, tendremos que definir las funciones en PHP, registrarlas y generar el código JavaScript necesario para disponer de ellas (mediante el método denominado **printJavascript**, que se ubica dentro del elemento HEAD de la página). Además, antes de cualquier salida, debemos incluir una llamada al método **processRequest**, que procesa y maneja las peticiones:

```

<?php

// funciones PHP

require_once("xajax.inc.php");

$xajax = new xajax();
$xajax->register(XAJAX_FUNCTION, "nombreFuncion");
$xajax->processRequest();

?>
<html>
<head> <?php $xajax->printJavascript(); ?> </head>
<body> ... </body>
</html>

```

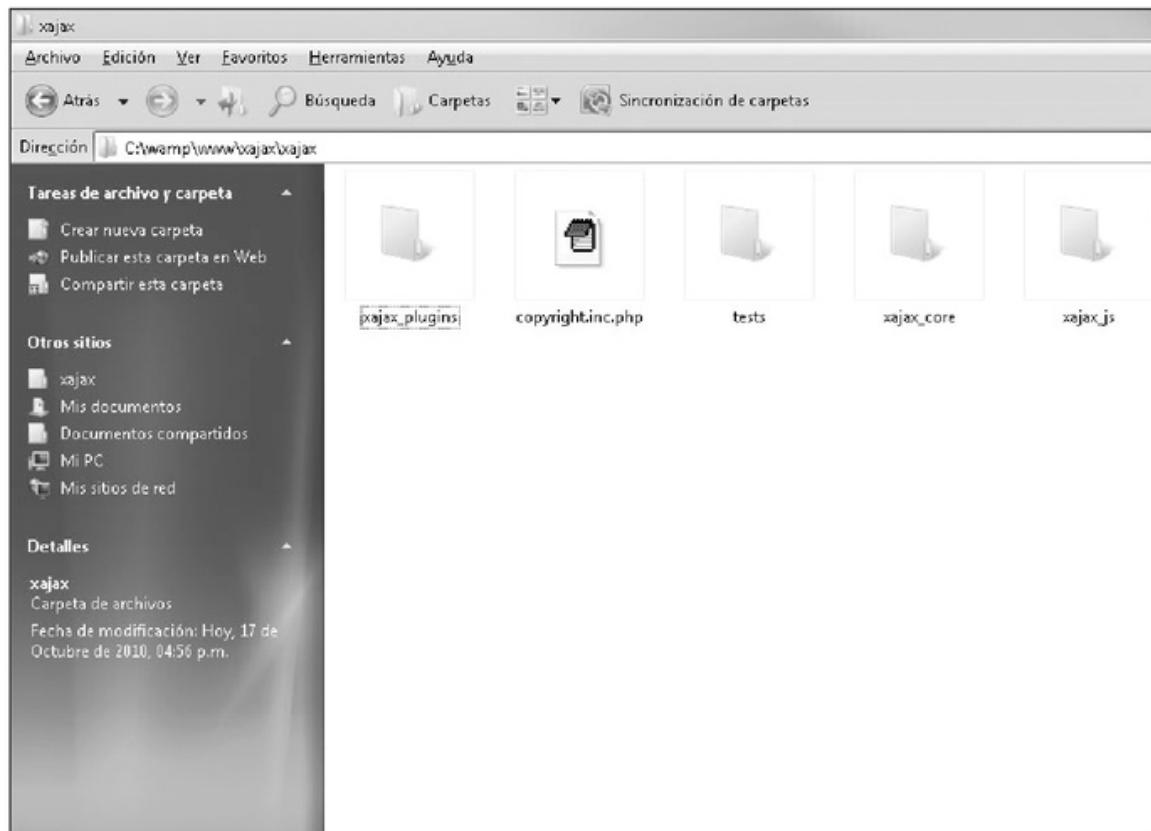


Figura 16. La distribución Xajax incorpora los archivos necesarios para desarrollar aplicaciones en PHP.

La sintaxis para invocar las funciones PHP desde el lado cliente se realiza anteponiéndoles el prefijo **xajax_**. Podemos modificar este comportamiento a través del método **configureMany**:

```
$xajax = new xajax();
$xconf['wrapperPrefix'] = 'nuevoPrefijo_';
$xajax->configureMany($xconf);
```

En el siguiente ejemplo, veremos cómo almacenar un contenido ingresado por el usuario en un archivo de texto. Primero definimos las funciones PHP que vamos a utilizar, llamadas **guardar** y **recuperar**:

```
function guardar($texto) {
    $respuesta = new xajaxResponse();

    if (file_put_contents('texto.txt', $texto))
        $TextoRespuesta = "Datos almacenados (" . date("H:i:s") . ")";
    else
        $TextoRespuesta = "Error al almacenar datos";

    $respuesta->assign("respuesta", "innerHTML", $TextoRespuesta);
    return $respuesta;
}

function recuperar() {
    $respuesta = new xajaxResponse();
    $respuesta->assign("texto", "value", file_get_contents('texto.txt'));
    return $respuesta;
}
```

Luego incluimos la clase principal de Xajax, instanciamos un objeto y registramos las funciones de la siguiente manera:

```
require_once("xajax/xajax_core/xajax.inc.php");

$xajax = new xajax();
$xajax->register(XAJAX_FUNCTION, "guardar");
$xajax->register(XAJAX_FUNCTION, "recuperar");
$xajax->processRequest();
```

Ya dentro del código HTML, invocamos el método **printJavascript** y ligamos la ejecución de la función **xajax_recuperar** al cargar la página:

```
<?php $xajax->printJavascript('xajax/'); ?>

<script language="javascript">
    window.onload = function() { xajax_recuperar(); };
</script>
```

Definimos un espacio para que el usuario edite el texto contenido en el archivo:

```
<table>
    <tr>
        <td><textarea id="texto" rows="10" cols="50"></textarea></td>
    </tr>
    <tr>
        <td><input type="button" value="guardar"
            onclick="xajax_guardar(document.getElementById('texto').value)"></td>
    </tr>
</table>
```

Y un elemento para mostrar la respuesta ante las acciones del usuario:

```
<div id="respuesta"></div>
```

La función **xajax_guardar** se ejecuta al presionar **guardar** para almacenar los datos.

Las funciones PHP correspondientes se definen y se registran. En las definiciones, vemos un objeto nuevo: **xajaxResponse**. Como dijimos, Xajax es orientado a objetos, y las respuestas también lo son. El método **assign** fija un valor (tercer argumento) a una propiedad (segundo argumento) de un elemento (identificado por el **ID** pasado en el primer argumento). Las respuestas tienen más métodos disponibles:

III LAS CAPAS DE AJAX

Ajax propone la división en capas de una aplicación, algo que va más allá del modelo y que se considera un principio de desarrollo general para casi cualquier sistema. **MVC** (*Model View Controller*) representa un antecedente concreto, ya que es una arquitectura que plantea la modularización de las funcionalidades en una aplicación.

MÉTODO	DESCRIPCIÓN
addEvent	Liga un evento a un elemento. Recibe como argumentos el ID del elemento, el evento (onclick, onmouseover, etcétera), y la acción por realizar (definida mediante código JavaScript).
addHandler	Similar a addEvent sólo que, como tercer argumento, recibe el nombre de la función JavaScript que se invocará al desatarse el evento.
addPrepend	Similar a append sólo que agrega el contenido antes del valor ya existente en la propiedad del elemento.
addReplace	Busca y reemplaza parte del valor de una propiedad. Recibe como argumentos el ID del elemento, la propiedad, el valor por buscar, y el valor con el que se reemplazará cada una de las ocurrencias del tercer argumento.
alert	Recibe como argumento un texto y lo muestra mediante la construcción alert de JavaScript.
append	Agrega un valor (tercer argumento) a una propiedad (segundo argumento) de un elemento (identificado por el ID pasado en el primer argumento). A diferencia de assign no reemplaza, sino que agrega un contenido luego del valor existente en la propiedad del elemento.
assign	Asigna un valor (tercer argumento) a una propiedad (segundo argumento) de un elemento (identificado por el ID pasado en el primer argumento).
call	Realiza una llamada a la función JavaScript incluida como primer argumento. Los parámetros a dicha función (si es que los tiene) se ubican en los sucesivos argumentos a call .
clear	Recibe como argumentos un elemento (identificado mediante un ID) y una propiedad, a la cual se le asigna una cadena vacía.
confirmCommands	Añade un pedido de confirmación a la respuesta.
contextAppend	Agrega al valor de un atributo de un objeto JavaScript otro valor (al final).
contextAssign	Modifica el valor de un atributo de un objeto JavaScript.
contextClear	Limpia el valor de un atributo de un objeto JavaScript.
contextPrepend	Agrega al valor de un atributo de un objeto JavaScript otro valor (al principio).
create	Crea un elemento. Recibe como argumentos el elemento contenedor, el tipo de elemento (por ejemplo div, h2, span, etcétera) y el nombre del elemento (el que será su ID).
createInput	Similar a create sólo que crea controles tipo input (text, radio, checkbox, etcétera). Recibe como argumentos el elemento contenedor, el tipo de elemento, el nombre del elemento (propiedad name) y el ID.
debug	Similar a alert .
getCommandCount	Devuelve el número de comandos retornado.
getContent-Type	Devuelve el content type retornado.
getXML	Devuelve el XML generado en la función PHP que será enviado al procesador Xajax. Sólo es útil a fines de investigación: Xajax envía la respuesta cuando encuentra la cláusula return en la función PHP.

MÉTODO	DESCRIPCIÓN
includeCSS	Incluye un archivo CSS. Recibe como argumento la ruta hacia él.
includeScript	Incluye un archivo JavaScript. Recibe como argumento la ruta hacia él.
includeScriptOnce	Similar a includeScript sólo que incluye el archivo una vez.
insert	Inserta un elemento antes de otro ya definido. Recibe como argumentos el ID del elemento existente, el tag del nuevo (h3, div, p, etcétera), y su ID.
insertAfter	Similar a insert sólo que inserta el nuevo elemento después de uno ya definido.
insertInput	Inserta un control input (text, radio, checkbox, etcétera) antes de un elemento ya existente. Recibe como argumentos el ID del elemento existente, el tipo del nuevo control, su propiedad name y su ID.
insertInputAfter	Similar a insertInput sólo que inserta el control después del elemento referenciado en el primer argumento.
outputEntitiesOff	No convierte los caracteres especiales incluidos en la respuesta a entidades HTML. Es el comportamiento por defecto.
prepend	Agrega un valor (tercer argumento) a una propiedad (segundo argumento) de un elemento (identificado por el ID pasado en el primer argumento). A diferencia de assign no reemplaza, sino que agrega un contenido antes del valor existente en la propiedad del elemento.
printOutput	Imprime la respuesta que será enviada al navegador.
redirect	Redirige la página actual a una pasada como argumento (como window.location).
remove	Recibe como argumento un ID de un elemento y lo remueve.
removeCSS	Remueve la referencia a un archivo CSS. Recibe como argumento la ruta hacia él.
removeHandler	Deshace la acción de addHandler . Recibe como argumento el ID del elemento, el evento y el nombre de la función JavaScript asociada.
removeScript	Remueve la referencia a un archivo JavaScript. Recibe como argumento la ruta hacia él.
replace	Busca un texto (tercer argumento) y lo reemplaza por otro (cuarto argumento) en una propiedad (segundo argumento) de un elemento (ID, primer argumento).
script	Añade y evalúa el código JavaScript incluido en el argumento.
setCharacterEncoding	Define el juego de caracteres utilizado en la respuesta. Si no se incluye de forma expresa, se toma el valor UTF-8, que figura en el archivo xajax.inc.php asignado a una constante.
setEvent	Similar a addEvent .
setFunction	Crea una función JavaScript. El primer argumento es el nombre de la función; el segundo, los argumentos separados por comas; el último, el cuerpo de función.
setOutputEntities	Convierte caracteres especiales a las entidades HTML correspondientes. Precisa que la extensión mb_string esté habilitada.
sleep	Espera n décimas de segundo antes de devolver el control.
waitFor	Espera hasta que el script pasado como argumento haya sido ejecutado.
waitForCSS	Espera hasta que el CSS haya sido cargado o removido.

Tabla 2. Métodos disponibles para *xajaxResponse*.



Figura 17. Xajax está orientado a toda clase de proyectos, desde los simples hasta los más complejos.

La opción de configuración **javascript URI** nos permite definir la ruta al directorio donde se encuentra la carpeta **xajax_js**:

```
$xajax->configure('javascript URI','./xajax');
```

Debemos recordar que si la función por registrar no se encuentra en el mismo archivo, podemos incluir un tercer argumento que contenga la ruta al archivo que la contiene en la llamada a **register**:

```
$xajax->registerFunction("recuperar", "funciones.php");
```

III DISPONIBILIDAD

Es importante saber que las aplicaciones Ajax no están orientadas a sistemas complejos o simples; pueden adecuarse a distintos ámbitos, y los casos de uso son de los más variados. La gran cantidad de opciones puestas a disposición del desarrollador por Xajax es una prueba de este hecho. Por esta razón, se convierte en una herramienta muy versátil para quien la utiliza.

La respuesta es un fragmento XML. En el caso del ejemplo anterior, podría ser:

- Respuesta para **xajax_recuperar**:

```
<?xml version="1.0" encoding="utf-8" ?>
<xjx>
<cmd n="as" t="texto" p="value">este es el texto</cmd>
</xjx>
```

- Respuesta para **xajax_guardar**:

```
<?xml version="1.0" encoding="utf-8" ?>
<xjx>
<cmd n="as" t="respuesta" p="innerHTML">Datos almacenados (21:50:42)</cmd>
</xjx>
```

Para configurar el tipo de petición (método **POST** o **GET**), podemos utilizar el método **configureMany**:

```
$xajax = new xajax();
$conf['defaultMethod'] = 'GET';
$xajax->configureMany($conf);
```

Xajax admite la opción de trabajar en modo debug:

```
$xajax = new xajax();
$conf['debug'] = true;
$xajax->configureMany($conf);
```

III PROTOTYPE

En este capítulo, mencionamos la librería denominada **Prototype** en diversas ocasiones: su calidad dentro de las muchas opciones disponibles para implementar Ajax en aplicaciones es imposible de negar y, por sus características, su facilidad de uso y su documentación, se posiciona como uno de los desarrollos recientes más importantes.

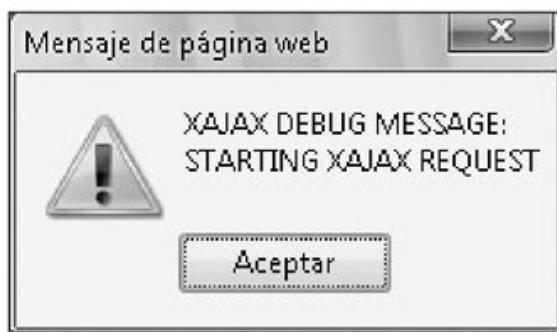


Figura 18. El modo debug en Xajax es importante a la hora de probar el funcionamiento de nuestras aplicaciones.

Este método admite otras posibilidades interesantes, listadas en **xajax.inc.php**, y podemos incluir más de una:

```
$xajax = new xajax();
$xconf['debug'] = true;
$xconf['defaultMethod'] = 'GET';
$xconf['allowBlankResponse'] = true;
$xconf['timeout'] = 3000;
$xconf['javascript URI'] = './xajax';
$xajax->configureMany($xconf);
```

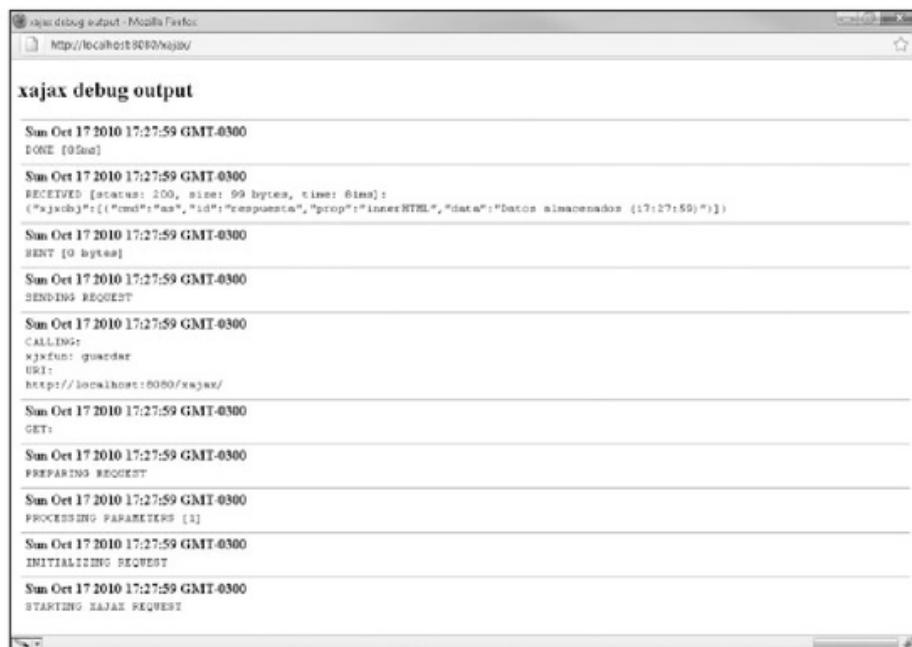


Figura 19. Xajax mantiene un principio de desarrollo claro que nos permite comprender el proceso cliente/servidor de Ajax.

El juego de caracteres ISO-8859-1 (o Latin-1) deriva del ISO-8859 y cuenta con caracteres usados comúnmente en idiomas como albanés, catalán, danés, alemán,

inglés, francés, finlandés, islandés, irlandés, italiano, latín, noruego, portugués, escocés, suizo y español. Por lo general, se utiliza cuando es necesario incorporar caracteres acentuados (sin el ´ de HTML) o la letra ñ (minúscula y mayúscula). La opción para modificar este valor es **characterEncoding**:

```
$conf['characterEncoding'] = 'UTF-8';
```

```
$conf['characterEncoding'] = 'ISO-8859-1';
```

jPOP

jPOP (*JavaScript Powered On PHP*) es una alternativa ideal para aquellos desarrolladores que quieran implementar Ajax en sus aplicaciones sin sobrecargar las páginas (la librería base pesa alrededor de 16 KB) ni gastar demasiado tiempo en realizar las configuraciones necesarias para que todo funcione. Su uso es libre, y podemos descargar la última versión disponible si visitamos la página que se encuentra en la dirección web <http://jpop-framework.sourceforge.net>.

The screenshot shows the SourceForge.net project page for "jPOP Framework". The left sidebar has sections for "Open Source Software" and "Sourceforge.net Hosted". The main content area has a header "jPOP Framework". It includes a "Google AdWords" advertisement, a "Project Information" section with "About this project" and "Project team" details, and a "Developers" section with links for "Join this project", "Get the source code", and "Update project web pages".

jPOP Framework

Open Source Software

Sourceforge.net Hosted

Users

- [Download jPOP Framework files](#)
- [Donate money](#)
- [Project detail and discuss](#)
- [Get support](#)

Not what you're looking for?

SourceForge.net hosts over 100,000 Open Source projects. You may find what you're looking for by [searching our site](#) or using our [Software Map](#).

You may also want to consider these similarly-categorized projects:

- [Boglipse](#)
- [Pidgin](#)
- [XUI RIA Framework](#)
- [OpenNMS](#)
- [PHP For Applications - PHP Framework](#)

Google Cuánto cuesta hacer public

Anunciá en Google AdWords

Mínima inversión máximos resultados. Pagás por clic. ¡Desde \$0,10!

www.PromoGoogleAdwords.com

Atraer clientes cuesta menos

Consegi clientes con Google AdWor

Project Information

About this project:

This is the **jPOP Framework** project ("jpop-framework")

This project was registered on SourceForge.net on Feb 16, 2007, and is described by the project team as follows:

jPOP is a simple but powerful Ajax and RIA framework for PHP. It requires absolutely no coding on the client-side. By minimizing effort and time, and maximizing elegance and readability of code, sleek web apps are

Developers

Join this project:

To join this project, please contact the project administrators of this project, as shown on the [project summary page](#).

Get the source code:

Source code for this project may be available as [downloads](#) or through one of the SCM repositories used by the project, as accessible from the [project develop page](#).

Update project web pages:

This page is the default project web page supplied by SourceForge.net. If you are a member of this project, you can deploy your own project web pages as per our [site documentation](#).

If you are a web page developer interested in this project, please consider reaching out to the project admin (per the "Join this project"

Figura 20. jPOP es una opción rápida y sencilla para incluir soluciones Ajax en los desarrollos.

Descomprimidos los archivos, copiamos en un directorio accesible por las demás páginas y configuramos el archivo **config.php**. La variable **path** indica la ruta hacia el directorio que contiene la raíz de la distribución jPOP el que incluye al archivo **jPOP.php**:

```
$path = './jPOP/';
```

Para disponer de las funcionalidades de esta librería, debemos ubicar en cada página la siguiente línea, suponiendo que jPOP se encuentra en un directorio llamado jPOP:

```
<?php require('jPOP/jPOP.php'); ?>
```

Y por último, generar el código JavaScript necesario a través del método **init**:

```
<head> <?php jPOP::init(); ?> </head>
```

jPOP posee tres funciones principales:

- **newAjaxControl**
- **newAjaxDiv**
- **newAjaxLink**

Mediante **newAjaxControl**, podemos crear un control que modificará asincrónicamente el contenido (propiedad **innerHTML**) de otro ya creado. Recibe como argumentos:

- El identificador del control (**Id** y **Name**).
- El valor del control (**Value**).
- El identificador del elemento destino (**Id**).
- Archivo que devolverá la respuesta (**pagina.php**, **archivo.txt**, etcétera).
- Evento que desatará la petición (**onClick**, **onMouseOver**, etcétera).
- Tipo de control, de manera predeterminada **button**. Opcional.
- HTML exhibido mientras se recupera la respuesta. Opcional.
- Efecto desatado al mostrar la respuesta. Opcional.

index.php

```
<?php require('jPOP/jPOP.php'); ?>
<html>
```

```

<head>
<?php jPOP::init(); ?>
</head>

<body>
<?php

newAjaxControl('nombreControl', 'Que hora es?', 'nombreContenedor',
    'respuesta.php', 'onClick', 'button');

?>

<div id="nombreContenedor" name="nombreContenedor" />

</body>
</html>

```

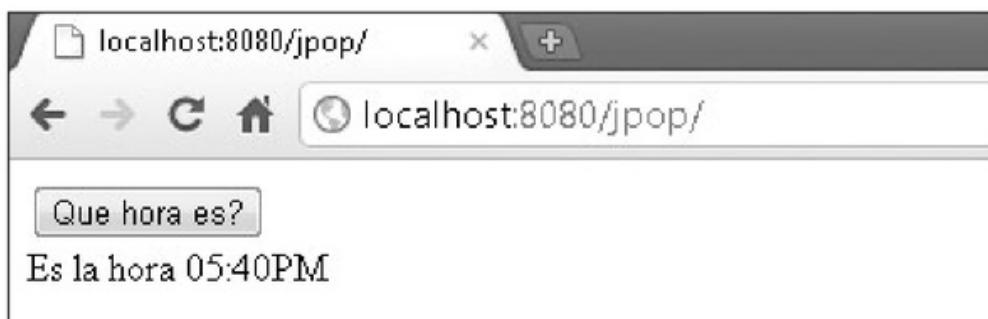


Figura 21. Con pocas líneas de código, a través de jPOP podemos generar peticiones al servidor.

respuesta.php

```
<?php echo "Es la hora ".date("h:iA"); ?>
```



PROCESOS SINCRÓNICOS

Es interesante constatar que en las aplicaciones web tradicionales, cuando un navegador se encarga de realizar una petición, la actividad del usuario se interrumpe momentáneamente hasta que el servidor envía la respuesta. Los procesos en el lado cliente y servidor son sincrónicos, es decir, no puede continuar uno si el otro no termina.

Como observamos, jPOP incluye Prototype en su distribución, permitiendo incluir los efectos y comportamientos disponibles. Prototype es un framework JavaScript de propósito general orientado a objetos, que podemos descargar desde su página oficial: www.prototypejs.org.

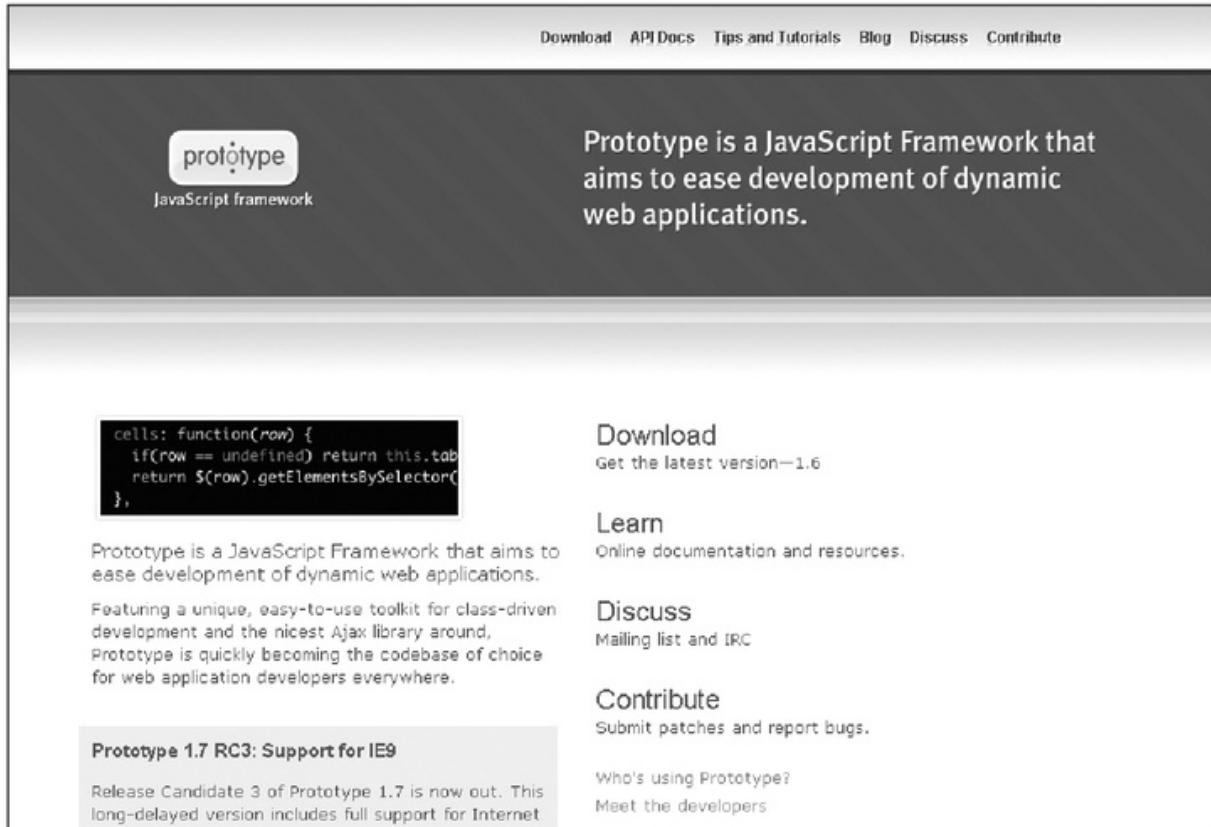


Figura 22. Prototype es un popular framework que enmascara el acceso a las funcionalidades provistas por JavaScript.

La función **newAjaxDiv** genera un elemento **DIV** capaz de actualizar otros elementos. Es similar a **newAjaxControl** y recibe como argumentos:

- El identificador del **DIV (Id y Name)**.
- Contenido original del **DIV (innerHTML)**.
- El identificador del elemento destino (**Id**, puede reemplazarse a sí mismo).
- Archivo que devolverá la respuesta (**pagina.php**, **archivo.txt**, etcétera).
- Evento que desatará la petición (**onClick**, **onMouseOver**, etcétera).
- HTML exhibido mientras se recupera la respuesta. Opcional.
- Efecto desatado al mostrar la respuesta. Opcional.

index.php

```
<?php require('jPOP/jPOP.php'); ?>
```

```

<html>
  <head>
    <?php jPOP::init(); ?>
  </head>

  <style>
    #nombreControl {
      cursor: pointer;
    }
  </style>

  <body>
    <?php

      newAjaxDiv('nombreControl', 'Que hora es?', 'nombreContenedor',
      'respuesta.php', 'onClick');

    ?>

    <div id="nombreContenedor" name="nombreContenedor" />

  </body>
</html>

```

respuesta.php

```
<?php echo "Es la hora ".date("h:iA"); ?>
```

Por su parte, la función **newAjaxLink** es en extremo parecida a **newAjaxDiv**, sólo que en lugar de generar una etiqueta de tipo **DIV** genera un enlace HTML. Debe recibir como argumentos:

- Contenido original del **DIV (innerHTML)**.
- El identificador del elemento destino (**Id**, puede reemplazarse a sí mismo).
- Archivo que devolverá la respuesta (**pagina.php**, **archivo.txt**, etcétera).
- Evento que desatará la petición (**onClick**, **onMouseOver**, etcétera).
- HTML exhibido mientras se recupera la respuesta. Opcional.
- Efecto desatado al mostrar la respuesta. Opcional.

index.php

```
<?php require('jPOP/jPOP.php'); ?>
<html>
  <head>
    <?php jPOP::init(); ?>

  </head>

  <style>
    font-family: Arial, Helvetica, sans-serif;
    font-size: 14px;
    font-weight: normal; body {
  }

  a {
    cursor: pointer;
    color: #3B3B3B;
    text-decoration: none;
  }

  #nombreContenedor {
    color: #CC3333;
  }
</style>

<body>
<?php

newAjaxLink('Que hora es?', 'nombreContenedor', 'respuesta.php',
  'onClick');

?>

<div id="nombreContenedor" name="nombreContenedor" />

</body>
</html>
```

respuesta.php

```
<?php echo "Es la hora ".date("h:iA"); ?>
```

Y el link generado:

```
<a onClick = 'callAjax( "nombreContenedor" , "respuesta.php" , "loading..." , "appear" )' >Que hora es?</a>
<div id="nombreContenedor" name="nombreContenedor" />
```

jPOP nos permite generar ventanas a partir de Prototype Window (<http://prototype-window.xilinus.com>) de manera sencilla. Para disponer de esta opción, simplemente tendremos que habilitar en el archivo **config.php** el plug-in correspondiente (**prototype_windows**):

```
$active_plugins = array(
    'UIControls',
    'prototype_windows'
);
```

Luego, generar en los scripts instancias de la clase **Window**, que recibe los siguientes argumentos, todos opcionales:

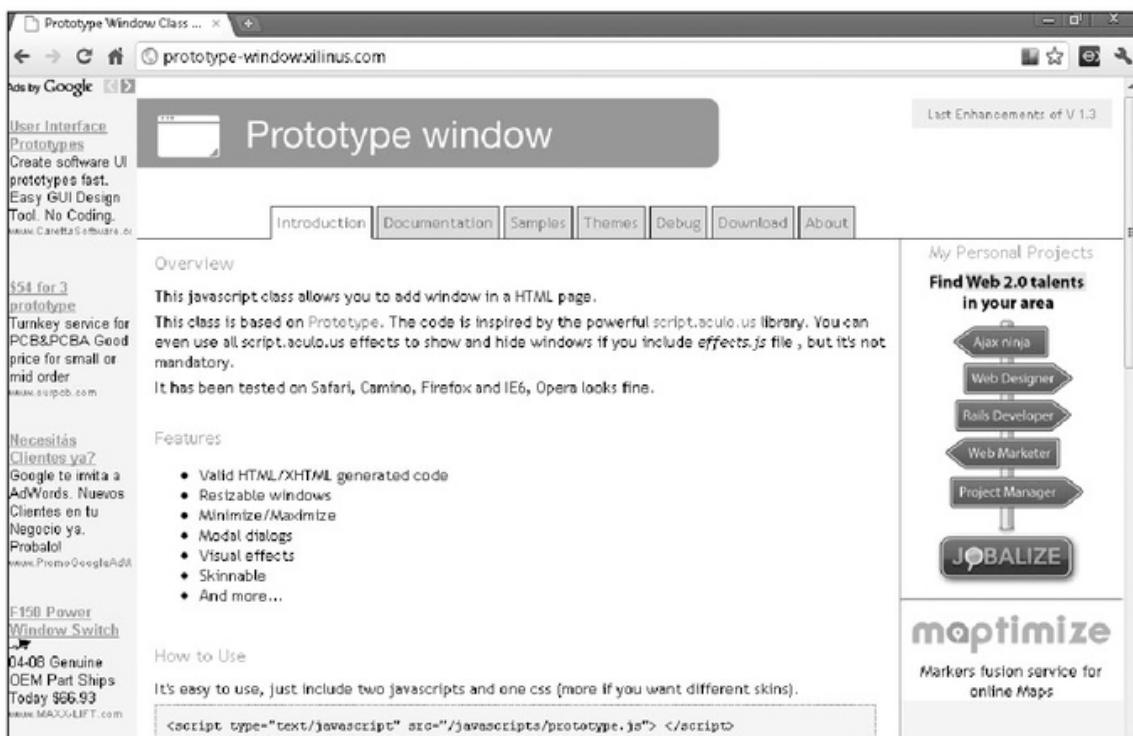


Figura 23. Prototype Window es otra de las librerías externas incluidas en jPOP.

- Texto de la ventana.
- Título.
- Ancho en pixeles.
- Alto en pixeles.
- Tamaño modificable o no (**true** o **false**).
- Efecto utilizado al abrir la ventana.
- Efecto utilizado al cerrar la ventana.
- Si se va a poder arrastrar o no (**true** o **false**).
- Otras opciones propias de Prototype Window.

index.php

```
<?php require('jPOP/jPOP.php'); ?>
<html>
  <head>
    <?php jPOP::init(); ?>
  </head>

  <body>
    <?php

$ventana = new Window("texto<br /> de la ventana", "titulo", "200", "200",
    true, 'Appear', 'Fade', true);
$ventana->show();

?>
  </body>
</html>
```

Como podemos observar, jPOP interactúa con herramientas externas de manera sencilla y se complementa para dar robustez y potencia a las aplicaciones generadas.

III TECNOLOGÍAS

El metalenguaje XML aparece formando parte del acrónimo Ajax como formato preferido para intercambiar datos con el servidor. Sin embargo, no es la única alternativa: por ejemplo, vimos cómo JSON puede ocupar su lugar. Recordemos que Ajax no se liga a ninguna tecnología en particular, dejando la elección en manos de los desarrolladores.



Figura 24. La generación de ventanas es una de las funcionalidades disponibles incluidas en jPOP.

A través de **showAjax**, es posible recuperar el contenido de la ventana desde un documento externo:

```
$ventana->showAjax('respuesta.php');
```

Podemos incluir elementos en las ventanas a partir de HTML plano o utilizando las clases provistas por el plug-in **UIControls**:

```
<?php require('jPOP/jPOP.php'); ?>
<html>
  <head>
    <?php jPOP::init(); ?>
  </head>

  <body>
    <?php

$ventana = new Window("texto de la ventana", "titulo", "200", "200", true,
'Appear', 'Fade', true);
```

```
$boton = new Button();
$boton->init("nombre", "valor");

$texto = new textBox();
$texto->init("nombre", "valor", 10);

$textarea = new TextArea();
$textarea->init("nombre", "valor");

$ventana->add("<br />");
$ventana->add("<br />");
$ventana->add($texto);
$ventana->add("<br />");
$ventana->add($textarea);
$ventana->add("<br />");
$ventana->add($boton);

$ventana->show();

?>
</body>
</html>
```

Además, jPOP permite la creación de plug-ins por parte del usuario.



RESUMEN

PHP es un lenguaje poderoso que se caracteriza por dar respuesta a la mayoría de las necesidades de los desarrolladores, y la implementación de soluciones basadas en Ajax no es la excepción. En este capítulo, incluimos una breve reseña acerca de las características del modelo de Ajax y observamos sólo tres de las muchas opciones disponibles hoy en día en el mercado para trabajar desde PHP: Sajax, Xajax, y jPOP.



ACTIVIDADES

PREGUNTAS TEÓRICAS

1 ¿En qué situaciones incluiría Ajax en sus desarrollos?

2 ¿Qué precauciones tomaría en materia de seguridad?

3 ¿En qué situaciones no incluiría Ajax en sus desarrollos?

4 ¿Cree que el usuario se sentiría cómodo utilizando aplicaciones Ajax?

5 Nombre tres aplicaciones web que utilicen Ajax.

6 ¿Qué tarea cumple el objeto XMLHttpRequest?

7 ¿Cuál es el soporte por parte de los navegadores actuales?

8 Nombre una de las diferencias entre Sajax y Xajax.

EJERCICIOS PRÁCTICOS

1 Implemente un sistema de búsqueda utilizando jPOP.

2 Desarrolle una galería de imágenes mediante Xajax.

3 Genere un sistema de pestañas (tabs) usando Sajax.

RAD en PHP

El desarrollo rápido de aplicaciones es ya una realidad para todo tipo de programadores gracias a herramientas como CakePHP tratada en este capítulo a través de ejemplos prácticos de sus diversos usos. Se incluye también la vinculación de CakePHP con bases de datos y las opciones disponibles para relacionar Modelos dentro de una aplicación de tipo MVC con PHP.

Características	134
CakePHP	135
Modelos	138
ABMC dinámicos	140
Relaciones entre modelos	144
Generadores	147
Templates	148
Ejemplo práctico de aplicación	150
Resumen	165
Actividades	166

CARACTERÍSTICAS

La experiencia de distintos usuarios ante esta clase de frameworks nos indica que el período de aprendizaje puede ser complejo al principio (llegar a conocer todas las funcionalidades de una herramienta de estas características no es inmediato), pero luego los beneficios comienzan a ser claramente perceptibles. Esta clase de aplicaciones basan su funcionamiento en el modelo **MVC** (*Model View Controller*), que básicamente divide una aplicación en capas, cada una con su propia función específica:



Figura 1. Desde el sitio oficial de CakePHP, podemos consultar las últimas novedades disponibles y descargar la versión más actualizada.

- Los **modelos** son representaciones de las tablas de una base de datos y se encargan de conectarlas o de ligarlas a una aplicación.
- Un **controlador** contiene la lógica de una aplicación. A partir de aquí, es posible recuperar la información desde una base de datos a través de la capa modelo.
- Las **vistas** contienen la presentación de las páginas y el código necesario para mostrar la información. La interacción entre los modelos y las vistas se da a través de los controladores.

La alta popularidad alcanzada por **Ruby on Rails** (que implementa MVC) y el avance de PHP en relación con el soporte brindado a la programación orientada a objetos (**POO**, desarrollaremos más información acerca de este paradigma en el **Capítulo 6**) establecieron el escenario ideal para que surgieran los frameworks para desarrollo rápido de aplicaciones.

CAKEPHP

Una de las alternativas más utilizadas del momento es, sin dudas, **CakePHP**, que mantiene una gran comunidad de usuarios y permite generar aplicaciones funcionales con sólo unas pocas líneas de código. Está disponible en www.cakephp.org, y desde allí podemos descargar la última versión estable. Además es libre y de código abierto.

Es importante remarcar que CakePHP puede trabajar con versiones PHP 4 y superiores, posibilidad que no todos los frameworks de este tipo ofrecen, ya que la mayoría precisa por lo menos la versión 5. Para funcionar, necesitaremos contar con un servidor HTTP (mejor si es Apache con soporte para sesiones y **mod_rewrite** habilitados), PHP 4.3.2 o superior, y un motor de bases de datos (MySQL, PostgreSQL, SQLite, o alguna con acceso a través de drivers provistos por PEAR y ADOdb) que actuará como repositorio para CakePHP.

Entre las características más importantes de este framework de Desarrollo Rápido de Aplicaciones, podemos citar las siguientes:

- Templates y caché para las vistas.
- Implementación de AJAX.
- *Scaffolding* (AMBCs dinámicos).
- Seguridad para control de acceso.
- URLs cortas y amigables.

Para comenzar con el uso de la herramienta, simplemente, debemos descomprimir el archivo de la distribución en una carpeta dentro del directorio raíz del servidor web. Al igual que en Ruby on Rails, CakePHP necesita para cada proyecto una estructura de directorios predefinida, por lo que es importante que mantengamos la organización original: cada directorio tiene una función específica y necesaria. Luego, podremos crear nuevos archivos y carpetas, y modificar sus contenidos para personalizar nuestras aplicaciones.

Los directorios iniciales con los que contamos son:

DIRECTORIO	DESCRIPCIÓN
app	Aquí se incluirán los archivos específicos de la aplicación.
cake	Librerías base de CakePHP.
docs	Documentación.
vendors	Se almacenan las librerías externas que necesite la aplicación.

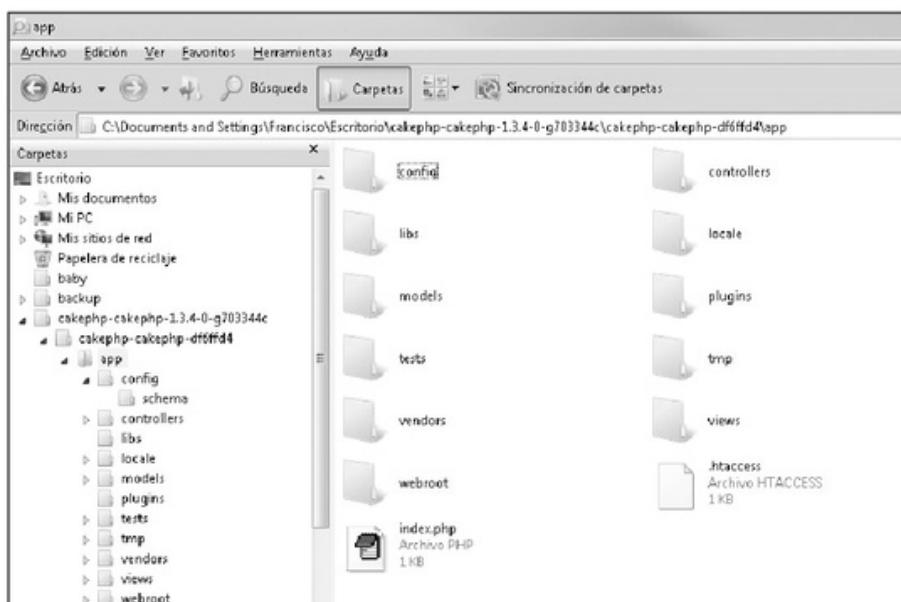
Tabla 1. Estructura de directorios de CakePHP

A su vez, dentro de **app**, contamos con los siguientes directorios:

DIRECTORIO	DESCRIPCIÓN
config	Archivos de configuración (bases de datos, rutas, etcétera)
controllers	Controladores.
models	Modelos.
plugins	Extensiones.
tmp	Archivos temporales.
vendors	Librerías externas.
views	Vistas.
webroot	Aquí se almacenan las hojas de estilo, imágenes, archivos, rutinas JavaScript, etcétera, a las que tendrá acceso el sitio (desde las vistas).

Tabla 2. Estructura de directorios de app.

CakePHP, al igual que Ruby on Rails, basa gran parte de su funcionamiento ligando nombres de tablas a nombres de directorios y archivos, cada uno de los cuales se localizarán en carpetas específicas. Esto nos permitirá la automatización de tareas, algo que nos ahorrará trabajo durante el proceso de desarrollo.

**Figura 2.** La estructura de directorios de CakePHP nos permite estandarizar los desarrollos y localizar errores de manera sencilla.

III MOD_REWRITE

Debemos tener en cuenta que si no contamos con soporte para **mod_rewrite**, podemos obtener información acerca de cómo habilitarlo en el manual de CakePHP (<http://book.cakephp.org/view/308/Installing-CakePHP>). Si aun así no lo logramos, en el archivo **app/config/core.php**, encontraremos alternativas para seguir adelante.

Luego deberemos editar el archivo **app/config/database.php.default** renombrándolo a **database.php** y completando con los datos correctos:

```
class DATABASE_CONFIG
{
    var $default = array(
        'driver' => 'mysql',
        'connect' => 'mysql_connect',
        'host' => 'localhost',

        'login' => 'root',
        'password' => '',
        'database' => 'cake',
        'prefix' => ''
    );

    var $test = array(
        'driver' => 'mysql',
        'connect' => 'mysql_connect',
        'host' => 'localhost',
        'login' => 'root',
        'password' => '',
        'database' => 'cake_test',
        'prefix' => ''
    );
}
```

Si tomamos en cuenta el código anterior, por lo menos la base de datos llamada **cake** deberá existir. Además de la base por defecto (**\$default**), podemos incluir otras (en el ejemplo **\$test**) para utilizarlas durante el desarrollo según sea necesario, por lo general para realizar pruebas en un servidor local. En cuanto a la estructura de una tabla, seguiremos algunas premisas:

- Los nombres, para nuestro modelo, deberán estar en plural (y en inglés), por ejemplo, **users**, **administrators**, **books**, etcétera.
- Las claves primarias deberán nombrarse como **id** (es el comportamiento por defecto; si no es así, tendremos que especificar necesariamente la clave en el modelo a través de la propiedad **\$primaryKey**).
- Los nombres de las claves foráneas deberán ser nombrados de la siguiente forma: **nombreTablaEnSingular_id** (por ejemplo **administrator_id**, **book_id**, etcétera).

Podemos obtener más información acerca de este comportamiento en el archivo `/app/config/inflections.php`.

Si alguna vez trabajamos con el framework de Ruby on Rails, esto nos parecerá familiar: se trata de una implementación del **ActiveRecord**, un patrón de diseño que permite automatizar ciertas tareas usuales en un desarrollo de software. En resumidas cuentas, nos da la posibilidad de ligar las tablas de una base de datos relacional con la lógica del modelo de una aplicación en objetos que interactúan entre sí. De esta manera, CakePHP (cuya implementación del **ActiveRecord** intenta ser lo más parecida posible a la de Ruby on Rails) puede automatizar, mediante convenciones, la manipulación básica de tablas tratándolas como objetos cuyos métodos podrían ser **insertarRegistro** o **eliminarRegistro**, por ejemplo.

Una vez configurado el archivo `database.php`, si accedemos al directorio de CakePHP a través de un navegador, podremos observar la página por defecto que indica si todo funciona correctamente, como vemos en la figura.



Figura 3. CakePHP está disponible tanto para PHP4 como para PHP5 y superiores, algo que lo diferencia de los demás frameworks.

Modelos

En CakePHP, cada modelo hace referencia a una tabla de una base de datos, y su nombre será el singular del de la tabla. Por ejemplo, el modelo **book** hará referencia a la tabla **books**, y a su vez estará implementado en el archivo `/app/models/book.php`. Dentro de la definición de un modelo, podremos definir reglas de negocio a través de métodos:

```
<?php
class Book extends AppModel {
//definicion
}

?>
```

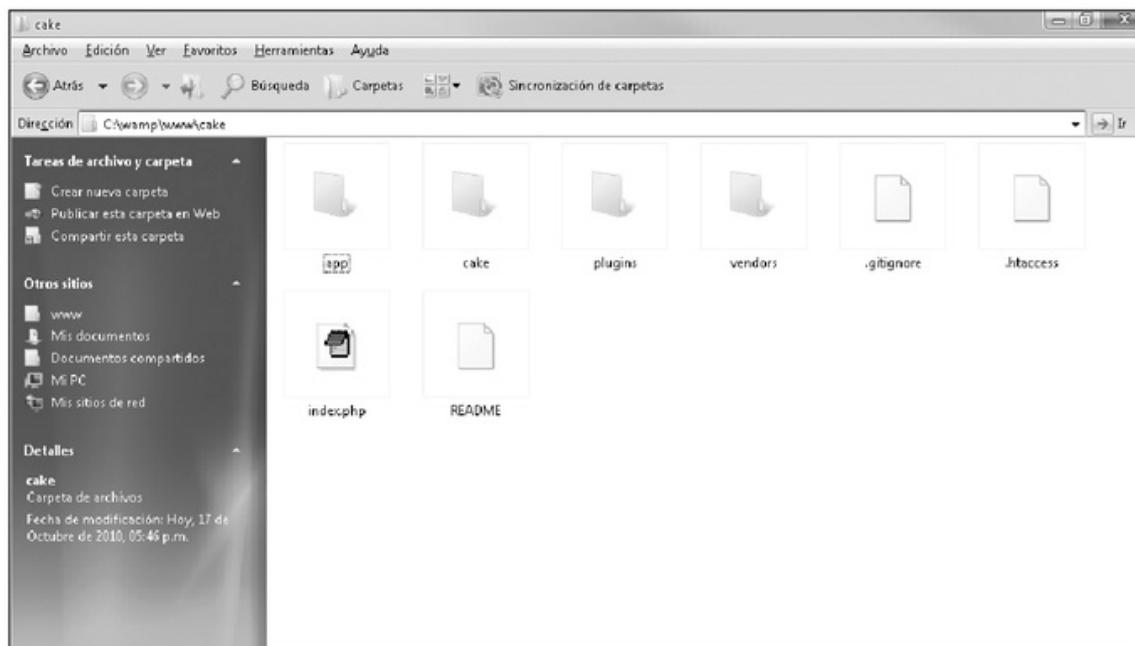


Figura 4. Comenzar el trabajo con CakePHP no supone demasiada dificultad: sólo debemos descargar la última versión de la herramienta y editar algunos archivos de configuración.

Modo de funcionamiento para resolver URLs

En el marco de una aplicación CakePHP, cuando el usuario intenta acceder a una sección, lo hace mediante la sintaxis:

```
http://servidor.com/nombreControlador/nombreMetodo
```

Por ejemplo:

```
http://servidor.com/messages/index
```

Lo primero que se hace es ejecutar el método **index** del controlador **messages**, definido en el archivo **messages_controller.php**. Este método invoca al modelo **Message** que recupera los datos necesarios (desde una base de datos, por ejemplo) y devuelve

el resultado al controlador. Éste a su vez pasa el control a la vista, que da un formato de salida a los datos recuperados. Los parámetros adicionales son argumentos al método.

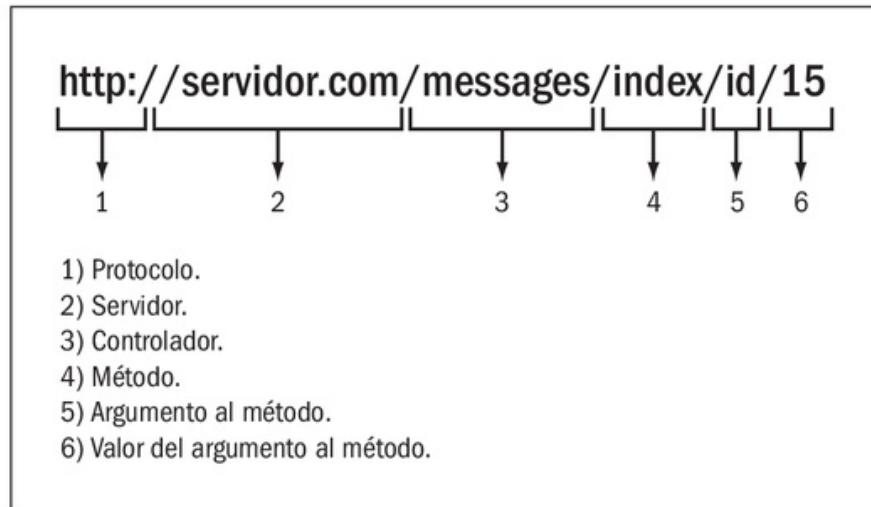


Figura 5. Las URLs tienen un significado importante para las aplicaciones basadas en CakePHP.

En CakePHP, las URL deberán tener correspondencia con una acción de un controlador. Para definir esta clase de comportamientos, deberemos editar el archivo `app/config/routes.php`, que tiene líneas como:

```
$Route->connect(ruta, array('controller' => 'nombreControlador', 'action' => 'nombreAccion', 'nombreParametro1', 'nombreParametroN'));
```

Esto es útil cuando precisamos redirigir a una determinada sección de un sitio de manera automática, como veremos más adelante.

ABMC dinámicos

CakePHP es capaz de generar **ABMC** (Alta Baja Modificación Consulta) o **CRUDs** (*Create Read Update Delete*) de registros, de manera automatizada. Si bien es probable

III CONFIGURACIÓN DEL FRAMEWORK EN CAKEPHP

La aplicación CakePHP consta de múltiples archivos para configurar el funcionamiento del framework, entre los más importantes podemos citar a `app/config/core.php`, que está compuesto por constantes comentadas acerca de diversos temas tales como debugging, el manejo de errores, caché, y también las sesiones, entre otros.

que necesitemos agregarles reglas particulares o modificar ciertas funcionalidades, una porción mayoritaria del trabajo estará hecho. Esto lo logramos declarando la variable `$scaffold` en el controlador. En el siguiente proyecto, tomamos como ejemplo una base de datos llamada `abmc`, cuya estructura es la siguiente:

```

CREATE DATABASE abmc;
USE abmc;

CREATE TABLE users (
    id int(11) NOT NULL auto_increment,
    nameUser varchar(255) default NULL,
    PRIMARY KEY  (id)
);

CREATE TABLE videos (
    id int(11) NOT NULL auto_increment,
    nameVideo varchar(255) default NULL,
    user_id int(11) NOT NULL,
    PRIMARY KEY  (id)
);

```

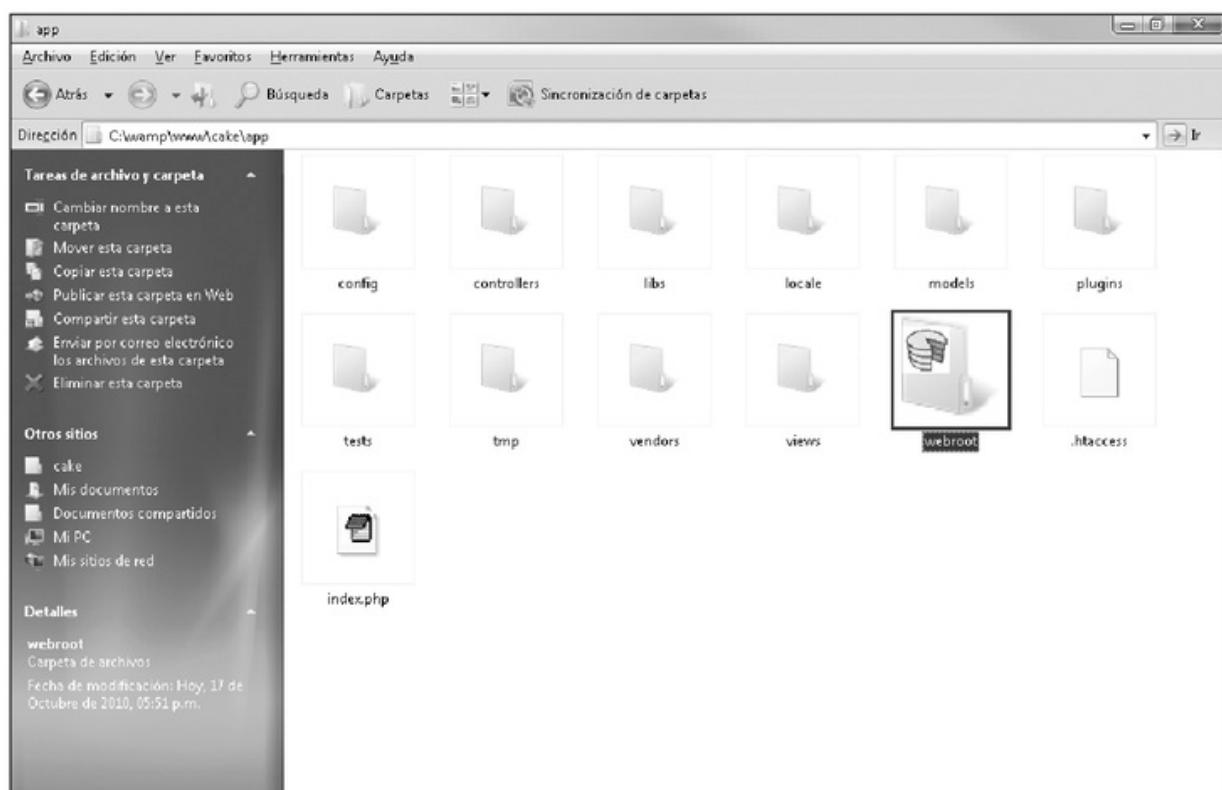


Figura 6. CakePHP facilita las tareas usuales en un desarrollo, permitiendo al programador enfocarse en los detalles específicos de la aplicación.

Primero editamos el archivo **app/config/database.php**, que debería verse similar a lo siguiente, si es que utilizamos como gestor a MySQL:

```
class DATABASE_CONFIG {
    var $default = array('driver' => 'mysql',
        'connect' => 'mysql_connect',
        'host' => 'localhost',
        'login' => 'root',
        'password' => '',
        'database' => 'abmc',
        'prefix' => '');
}
```

Comenzamos definiendo los modelos:

- **app/models/user.php**

```
<?php

class User extends AppModel {
    var $name = 'User';
    var $hasMany = array('Video');
}

?>
```

- **app/models/video.php**

```
<?php

class Video extends AppModel {
    var $name = 'Video';
    var $belongsTo = array('User');
}

?>
```

Tanto **hasMany** (**tiene muchos**) como **belongsTo** (**pertenece a**) nos sirven para relacionar modelos. Veremos más acerca de esto en las próximas páginas.

Una vez generados los modelos necesarios, continuamos con los controladores:

- **app/controllers/users_controller.php**

```
<?php

class UsersController extends AppController {
    var $name = 'Users';
    var $scaffold;
}

?>
```

- **app/controllers/videos_controller.php**

```
<?php

class VideosController extends AppController {
    var $name = 'Videos';
    var $scaffold;
}

?>
```



Figura 7. El *scaffolding* de CakePHP es una muestra de la relación del framework con Ruby on Rails, que también incluye una funcionalidad similar.

Y eso es todo. Si accedemos a **<http://localhost/cake/videos>** o a **<http://localhost/cake/users>** (suponiendo que **cake** es el directorio base de la aplicación), podremos visualizar y modificar registros. Por supuesto, en la mayoría de los desarrollos, esto no alcanza, pero nos sirve de muestra para observar las capacidades del framework.

Relaciones entre modelos

CakePHP nos permite relacionar modelos con el objetivo de representar fielmente las relaciones entre las tablas de una base de datos relacional. Las opciones disponibles son las siguientes:

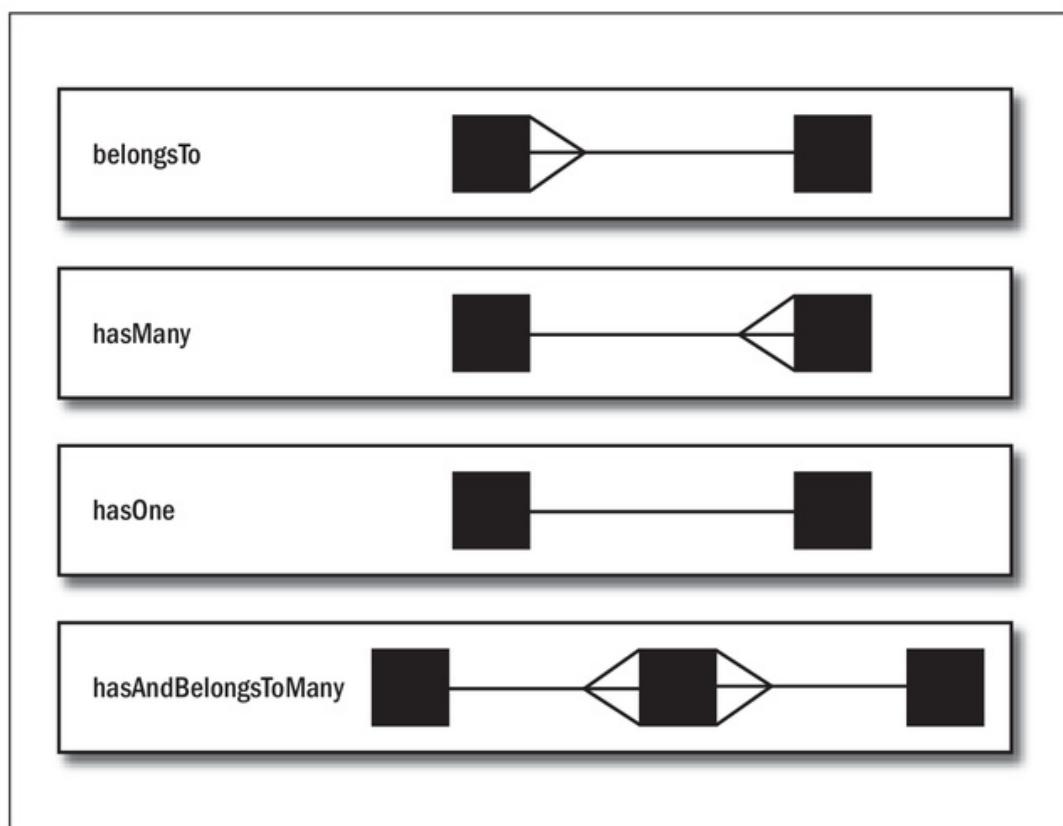


Figura 8. CakePHP permite exportar de manera intuitiva las relaciones entre entidades de una base de datos al ámbito de una aplicación.



MODELOS EN CAKEPHP

Podemos obtener una referencia completa acerca de la definición de la clase **AppModel** en el sitio web oficial de CakePHP, más precisamente en la sección **<http://api.cakephp.org/class/app-model>**. Allí podremos observar claramente y en detalle el comportamiento de los modelos en el contexto de una aplicación CakePHP.

- **belongsTo (pertenece a)**: cuando una tabla **A** tiene un campo que hace referencia a una tabla **B**, se dice que **A** pertenece a **B**. En una tabla que almacena mensajes y que tiene un campo que representa el código del autor, cada mensaje pertenece a un autor (tabla **autores**, por ejemplo).

```
class A extends AppModel {
    var $name = 'A';
    $belongsTo = array('B'=>array('className'=>'B'));
}
```

- **hasMany (tiene muchos)**: similar a **belongsTo**, sólo que observado desde el punto de vista de **B**. Es para relaciones uno a muchos. Si una tabla **A** tiene un campo que referencia a una tabla **B**, y más de un registro puede apuntar a **A**, se dice que **B** tiene muchos **A** (cada autor puede tener más de un mensaje asociado), ofrece el comportamiento inverso al descripto por **belongsTo**.

```
class B extends AppModel {
    var $name = 'B' ;
    $hasMany = array('A'=>array('className'=>'A'));
}
```

- **hasOne (tiene uno)**: es para las relaciones sencillas del tipo uno a uno. Si sólo un registro en una tabla **B** tiene un campo que hace referencia a una tabla **A**, se dice que **A** tiene un **B**, este método retornará un valor verdadero.

```
class A extends AppModel {
    var $name = 'A' ;
    $hasOne = array('B'=>array('className'=>'B'));
}
```



COMPATIBILIDAD

Como CakePHP está disponible para las versiones 4 y superiores del lenguaje de programación PHP, las clases, métodos y propiedades tomadas, puestas como ejemplo, no incluyen los modificadores **public**, **private**, etcétera, para mantener la compatibilidad. Veremos más acerca de la orientación a objetos en el **Capítulo 6**.

```
class B extends AppModel {
    var $name = 'B' ;
    $hasOne = array('A'=>array('className'=>'A'));
}
```

- **hasAndBelongsToMany** (también conocido como **HABTM**, **tiene y pertenece a muchos**): es para relaciones de muchos a muchos. Cuando una tabla **A** puede apuntar a más de un registro en **B**, y viceversa, se dice que **A** tiene y pertenece a muchos **B**, y que **B** tiene y pertenece a muchos **A**. Luego de la etapa de normalización, estas tablas son separadas por una tercera (**C**), que contiene por lo menos dos campos (uno que apunta a **A** y otro, a **B**).

```
create table discs (
    id int(11) primary key not null,
    title varchar(255),
    nameGroup varchar(255)
);
```

```
create table categories (
    id int(11) primary key not null,
    description varchar(255)
);
```

```
create table categories_discs (
    id int(11) primary key not null,
    disc_id int(11) not null,
    category_id int(11) not null
);
```

```
class Disc extends AppModel {
```

III NOMBRES DE COLUMNAS ESPECIALES

Si creamos dentro de una tabla de una base de datos una columna llamada **created** o **modified**, CakePHP las completará automáticamente en cada instrucción **insert** o **update** con la fecha de creación o de la última modificación del registro, según corresponda. Esto nos puede ser de utilidad para mantener un control sobre los cambios realizados.

```

var $name = 'Disc' ;

var $hasAndBelongsToMany = array('Categories' => array( 'className' =>
    'Category' ));
}

```

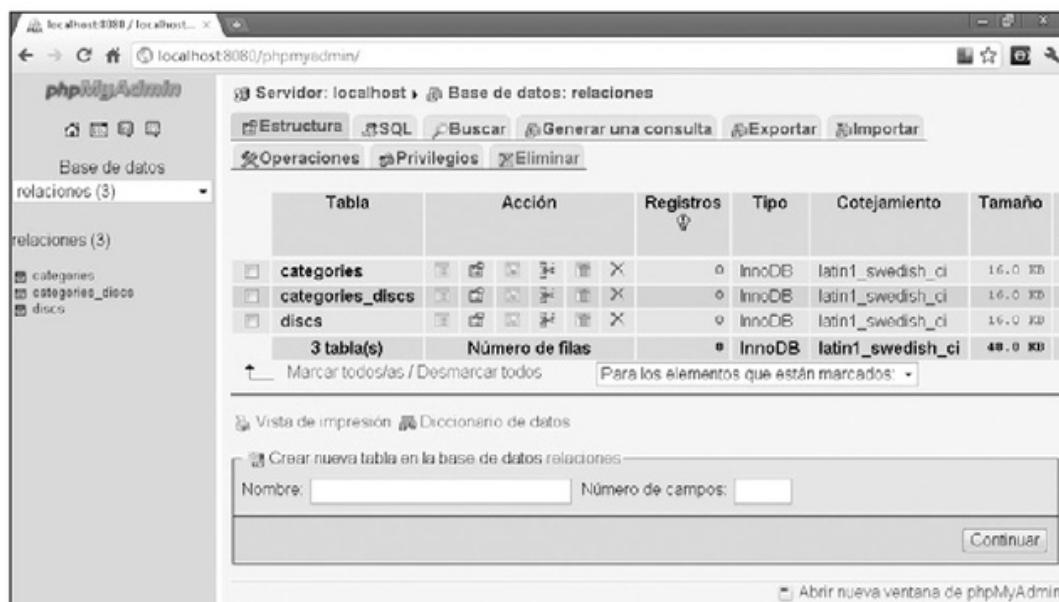


Figura 9. Las relaciones implementadas en CakePHP nos permiten transportar fielmente el modelo relacional a la lógica del framework.

```

class Category extends AppModel {
    var $name = 'Category';

    $hasAndBelongsToMany = array('Discs' => array( 'className' => 'Disc',
        'joinTable' => 'categories_discs'));
}

```

Generadores

CakePHP nos permite generar elementos HTML a través de los **View Helpers**, de la forma `$html->nombreMetodo`. Veamos algunos ejemplos de implementación:

```

<?php

echo $html->formTag('/messages/add/','post');

```

```

echo $html->image('imagen.jpg', array("alt"=>"texto alt", "title"=>"texto
title"));

echo $html->link('texto link', '/messages/index', array("class"=>
claseCss"), false, false, null);

echo $html->labelTag('Message/title', 'Texto label');

echo input('Message/content', array("size" => 20, "maxlength" => 20,
"class"=>"claseCss"));

echo $html->tagErrorMsg('Message/content', 'Mensaje error -
Campo obligatorio');

echo submit('Enviar', array("class"=>"claseCss"));

echo '</form>';
?>

```

El método **tagErrorMsg** nos permite mostrar mensajes de advertencia ante un ingreso incorrecto por parte del usuario.

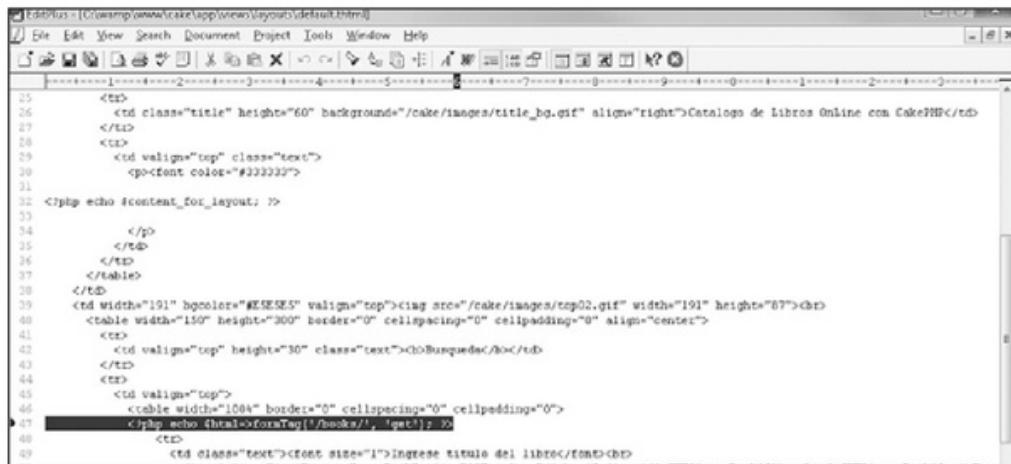


Figura 10. Si bien los **View Helpers** pueden omitirse y utilizar la sintaxis normal de los elementos, facilitan en gran medida las tareas de diseño.

Templates

Cada sitio puede contar con un template general para todas sus páginas. Este archivo se encuentra en **app/views/layout/default.thtml** y algunas de las funcionalidades añadidas que podemos definir son:

- Inclusión de hojas de estilos CSS. En el ejemplo, el archivo **nombreHojaDeEstilos.css** debe residir en **app/webroot/css/**:

```
$html->css('nombreHojaDeEstilos');
```

- Título de la página (elemento **title** del HTML):

```
$title_for_layout;
```

- Resultado de la vista:

```
$content_for_layout;
```

- Inclusión de archivos externos. En el ejemplo, el template **nombreInclude.thtml** debe estar en **/app/views/elements/**:

```
$this->renderElement('nombreInclude');
```

- Podemos definir desde cada controlador otro template base en lugar de **default.thtml**, que es el archivo que se utiliza de manera predeterminada:

```
$this->layout = 'noticias';
```

En el ejemplo anterior, se invoca al archivo **noticias.thtml**.

La extensión por defecto para los archivos que contienen las vistas es **THTML** (las si-glas corresponden a **Template HTML**). De manera similar, si trabajamos alguna vez con Ruby recordaremos que las plantillas tenían la extensión **RHTML (Ruby HTML)**.

III VIEW HELPERS

Es interesante saber que el uso de este tipo de generadores de tags HTML se encarga de permitirnos facilitar la codificación y por lo tanto mantener un control más exhaustivo sobre las vistas. Además, el código generado es XHTML, lo que admite realizar la posterior vali-dación de las páginas como bien formadas.

EJEMPLO PRÁCTICO DE APLICACIÓN

Para que podamos observar en funcionamiento algunas de las características de CakePHP, incluimos a continuación un proyecto para editar y visualizar libros y autores. Emplearemos MySQL más que nada por su popularidad y su simplicidad. Destaquemos que CakePHP puede vincularse con otras alternativas de manera sencilla, siempre y cuando se respete la sintaxis en los nombres de tablas y columnas. La base de datos que utilizaremos tiene la siguiente estructura:

```

CREATE DATABASE book;
USE book;

CREATE TABLE authors (
    id int(11) NOT NULL auto_increment,
    nameAuthor varchar(255) default NULL,
    lnameAuthor varchar(255) default NULL,
    PRIMARY KEY  (id)
);

CREATE TABLE books (
    id int(11) NOT NULL auto_increment,
    nameBook varchar(255) default NULL,
    descBook text,
    isbnBook varchar(255) default NULL,
    priceBook varchar(255) default NULL,
    author_id int(11) NOT NULL,
    PRIMARY KEY  (id)
);

insert into authors values (null, 'julio', 'cortazar');
insert into authors values (null, 'gabriel', 'garcia marquez');

```

III IMPLEMENTACIÓN DEL MODELO MVC

Debemos tener en cuenta que el popular Ruby on Rails toma los lineamientos definidos en la arquitectura MVC que, a través de modelos, vistas y controladores, garantiza la separación de la parte lógica por un lado y la de presentación por otro. Los frameworks RAD de PHP toman como base al modelo utilizado por Ruby on Rails.



Figura 11. La utilización de los nombres de las tablas se da en múltiples secciones de una aplicación CakePHP.

El primer paso será configurar el acceso a datos, para lo cual deberemos editar el archivo **app/config/database.php**:

```
class DATABASE_CONFIG {
    var $default = array('driver' => 'mysql',
        'connect' => 'mysql_connect',
        'host' => 'localhost',
        'login' => 'root',
        'password' => '',
        'database' => 'book',
        'prefix' => '');
}
```

Para ligar la página de inicio a la acción **index** del controlador **books**, editamos el archivo **app/config/routes.php**:

```
$Route->connect ('/', array('controller'=>'books', 'action'=>'index'));
```



Figura 12. CakePHP permite recuperar información desde una base de datos a través de los modelos relacionados.

A continuación, creamos e inicializamos los modelos (libros y autores):

- **book.php**

```
<?php

class Book extends AppModel {
    var $name = 'Book';

    var $belongsTo = array('Author' => array('className' => 'Author'));

    var $validate = array(
        'nameBook' => VALID_NOT_EMPTY,
        'descBook' => VALID_NOT_EMPTY,
        'isbnBook' => VALID_NOT_EMPTY,
        'priceBook' => VALID_NOT_EMPTY,
        'author_id' => VALID_NOT_EMPTY
    );
}

?>
```

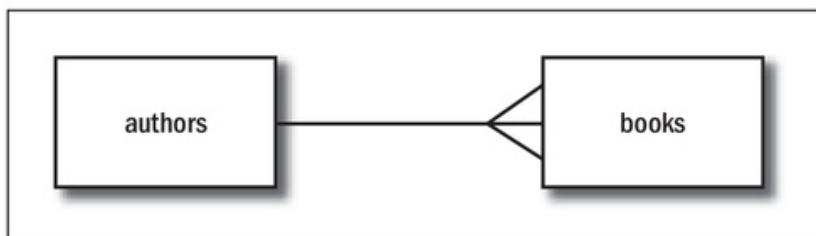


Figura 13. Las relaciones entre modelos se establecen en las aplicaciones CakePHP con unas pocas líneas de código.

- **author.php**

```

<?php

class Author extends AppModel {
    var $name = 'Author';

    var $hasMany = array('Book' => array('className' => 'Book'));

    var $validate = array(
        'nameAuthor' => VALID_NOT_EMPTY,
        'lnameAuthor' => VALID_NOT_EMPTY
    );
}

?>
  
```

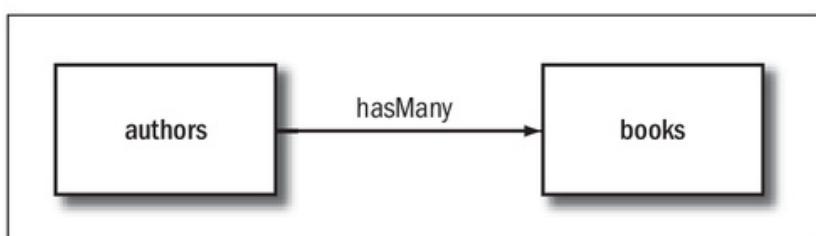


Figura 14. CakePHP mantiene una correspondencia con el modelo relacional de bases de datos en múltiples ámbitos, algunos de ellos son las relaciones entre entidades.

El array especial **validate** indica qué campos serán restringidos al momento de editar o de agregar nuevos registros. Hay varias clases de limitaciones posibles, enumeradas por constantes, entre las que se encuentran:

- **VALID_NOT_EMPTY** (no vacíos).
- **VALID_EMAIL** (email válido).
- **VALID_NUMBER** (numérico).

- **VALID_YEAR** (año válido).
- Expresiones regulares.



Figura 15. La validación de campos se puede detallar fácilmente a partir de las constantes puestas a disposición del desarrollador.

En cuanto a los controladores, el archivo **books_controller.php** incluye las sentencias necesarias para listar, visualizar, agregar, modificar y eliminar registros:

```
class BooksController extends AppController {
    var $name = 'Books';

    //métodos
}
```

Al ingresar en la pantalla inicial, podremos observar un listado de los libros disponibles, para lo cual se invoca al método **index**:

```
var $authors = '';
var $conditions = '';
var $fields = '';
var $order;
```

```

function index() {
    $this->pageTitle = 'Registros disponibles';

    if (isset($_GET['nameBook'])) {
        $conditions = array(" nameBook LIKE '%$_GET[nameBook]%'");
        $fields = array("*");
    } else {
        $conditions = $fields = '';
    }

    $order = "nameBook ASC";

    $this->set('books', $this->Book->findAll($conditions, $fields, $order));
}

```

Si accedemos a la opción para visualizar el detalle de una obra, el método que se invoca es **view**, lo haremos de la siguiente manera:

```

function view($id = null) {
    $this->pageTitle = 'Visualizacion de registros';
    $this->Book->id = $id;
    $this->set('book', $this->Book->read());
}

```

Para agregar nuevos registros a la tabla **books** de la base de datos **book**, contamos con el método **add**, definido de la siguiente manera:

```

function add() {
    $this->pageTitle = 'Agregar nuevo registro';
}

```



SYMFONY

Es importante recordar que otra interesante opción para implementar el desarrollo rápido de aplicaciones con el lenguaje PHP está dada por la utilidad **Symfony**, se trata de una herramienta cada vez más popular que trabaja con PHP versión 5 y superiores. Podemos encontrar más información en el sitio web oficial www.symfony-project.com.

```

if (!empty($this->data)) {
    if ($this->Book->save($this->data)) {
        $this->flash('Datos almacenados ! Volver al listado.', '/books');
    }
}

$this->setAuthors();
}

```

El método **flash** genera una página que tiene como contenido un mensaje (con el texto del primer argumento) que apunta a la dirección dada en el segundo argumento.

Por su parte, el método **setAuthors** carga los autores en una lista desplegable, permitiéndonos seleccionar el autor del nuevo libro:

```

function setAuthors() {
    $authors = $this->Book->Author->findAll();

    if (is_array($authors)) {
        foreach ($authors as $author) {
            $selectAuthor[$author['Author']['id']] = $author['Author']
                ['nameAuthor'] . ' ' . $author['Author']['lnameAuthor'];
        }
    }

    $this->set('authors', $selectAuthor);

    if (count($_POST)) {
        $this->set('selectedAuthor', $_POST['data']['Book']['author_id']);
    } else {
        $this->set('selectedAuthor', $this->data['Book']['author_id']);
    }
}

```

El mismo método se utiliza también para editar las obras ya existentes, para lo cual contamos con el método **edit**:

```
function edit($id = null) {
```

```

$this->pageTitle = 'Editar registros';

if (empty($this->data)) {
    $this->Book->id = $id;
    $this->data = $this->Book->read();

    $this->setAuthors();
} else {
    if ($this->Book->save($this->data['Book'])) {
        $this->flash('Datos almacenados ! Volver al listado.', '/books');
    } else {
        $this->setAuthors();
    }
}
}

```

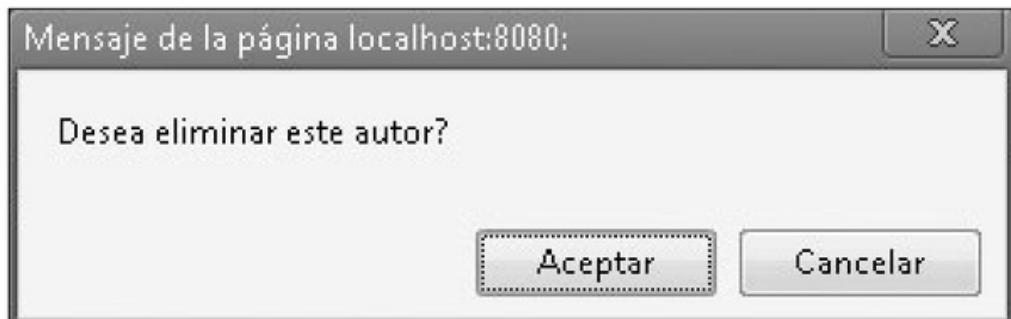


Figura 16. Las acciones comunes, como eliminar, agregar o modificar registros, son automatizadas por CakePHP.

El último método de los disponibles en el controlador **BooksController** es el que permite eliminar libros, llamado **delete**:

```

function delete($id) {
    $this->Book->del($id);
    $this->flash('Datos eliminados ! Volver al listado.', '/books');
}

```

El método **findAll** nos permite recuperar todos los registros y admite la posibilidad de restringir la selección, como se ve en el método **index** del controlador de libros:

```
$this->Book->findAll($conditions, $fields, $order)
```

Además contamos con otras opciones para obtener información:

MÉTODO	DESCRIPCIÓN
find	Recupera sólo una fila.
field	Recupera sólo un campo.
generateList	Se utiliza cuando en la vista necesitan generarse las opciones de una lista desplegable a partir de una tabla.
findAllBy	Recupera todos los registros a partir del nombre de un campo.
findBySQL	Recupera registros a partir de consultas SQL directas.

Tabla 3. Métodos del modelo para recuperación de datos.



Figura 17. El método findAll es utilizado en las búsquedas para recuperar registros que cumplan con una determinada restricción.

El controlador para autores (**authors_controller.php**) mantiene sus propios métodos para administrar registros, como podemos observar a continuación:

```
class AuthorsController extends AppController {
    var $name = 'Authors';

    //métodos
}
```

Para imprimir en pantalla el listado de autores disponibles, contamos con el método **index**, el cual es invocado inicialmente si es que no se especifica otra opción:

```
function index() {

    $this->pageTitle = 'Autores disponibles';
    $this->set('authors', $this->Author->findAll());
}
```

Para agregar nuevos autores (completando los campos **nombre** y **apellido**), disponemos del método **add**, aquí ejemplificado:

```
function add() {
    $this->pageTitle = 'Aregar nuevo registro';

    if (!empty($this->data)) {
        if ($this->Author->save($this->data)) {
            $this->flash('Datos almacenados ! Volver al listado.', '/authors');

        }
    }
}
```

En caso de eliminar un registro de la tabla **authors** de la base de datos **book**, el método que debemos invocar es **delete**:

```
function delete($id) {
    $this->Author->del($id);
    $this->flash('Datos eliminados ! Volver al listado.', '/authors');
}
```

Por último, el método **edit** nos permite modificar los datos de un autor y mostrar un mensaje referido a ello, como se ve en el siguiente código:

```
function edit($id = null) {
    $this->pageTitle = 'Editar registros';

    if (empty($this->data)) {
        $this->Author->id = $id;
        $this->data = $this->Author->read();
    } else {
        if ($this->Author->save($this->data['Author'])) {
            $this->flash('Datos almacenados ! Volver al listado.', '/authors');

        }
    }
}
```

En cuanto a las vistas, contaremos con un template base ubicado dentro de **views/layouts** llamado **default.thtml**, en el cual incluiremos un título:

```
<title><?php echo $title_for_layout; ?></title>
```

Como hoja de estilos, utilizamos **styles.css**, la cual es referenciada dentro del template mediante el método **css**:

```
<?php echo $html->css('styles'); ?>
```

Dentro del directorio **app**, existe otro llamado **webroot**, en donde se ubican los archivos del sitio, por ejemplo, las hojas de estilo (en la carpeta **css**).

Otra opción que agregamos al template son los enlaces para ingresar a ver los libros y autores disponibles (en ambos casos se accede por defecto al método **index**):

```
<?php echo $html->link('Libros', '/books/', array("class"=>"menu"), false,
false, null); ?>
```

```
<?php echo $html->link('Autores', '/authors/', array("class"=>"menu"),
false, false, null); ?>
```

La variable especial **\$content_for_layout** incluye el contenido de la vista actual, que dependerá del controlador y del método invocados:

```
<?php echo $content_for_layout; ?>
```

Por último, dentro de **default.thtml**, incluimos un formulario para realizar búsquedas de obras ingresando el título del libro:

```
<?php echo $html->formTag('/books/', 'get'); ?>
<table width="100%" border="0" cellspacing="0" cellpadding="0">

<tr>
<td class="text"><font size="1">Ingrese titulo del libro</font><br>
<input type="text" name="nameBook" size="18" value="<?php if
(isset($_GET['nameBook'])) echo $_GET['nameBook']; ?>">
```

```

</td>
</tr>
<tr>
    <td height="38">
        <input type="image" border="0" src="/cake/images/btn_submit.gif"
               width="58" height="19">
    </td>
</tr>
</table>
</form>

```



Figura 18. Si bien es posible incluir sentencias SQL en los controladores, para respetar la arquitectura MVC no se recomienda hacerlo.

Entre las vistas asociadas a las acciones de cada controlador, podemos citar, a modo de ejemplo, los que permiten editar libros y listar autores.

En el primer caso, dentro de **views/books/edit.thtml**, estará ubicado un formulario que apuntará a **/books/edit/**:

```

<form method="post" action="php echo $html-&gt;url('/books/edit/')?">
    <?php echo $html->hidden('Book/id'); ?>
    //contenido

```

```
</form>

<br /><?php echo $html->link("Volver al listado", "/books"); ?>
```

En su interior, encontraremos una tabla para ingresar nuevos valores en los distintos campos que componen la tabla **books**:

```
<table border="0" cellspacing="0" cellpadding="5">
<tr>
<th align="left" colspan="2"><h1>Editar libro</h1></th>
</tr>

//campos

</table>
```

Para definir el campo **isbnBook** de la tabla **authors**:

```
<tr>
<td bgcolor="#000000" class="filaTitulo">
<?php echo $html->tagErrorMsg('Book/isbnBook', '*') ?>ISBN
</td>
<td bgcolor="#c0c0c0">
<?php echo $html->input('Book/isbnBook', array('size' => '40'))?>
</td>
</tr>
```

Para definir el campo **nameBook** de la tabla **authors**:

```
<tr>
<td bgcolor="#000000" class="filaTitulo">
<?php echo $html->tagErrorMsg('Book/nameBook', '*') ?>Titulo
</td>
<td bgcolor="#c0c0c0">
<?php echo $html->input('Book/nameBook', array('size' => '40'))?>
</td>
```

```
</tr>
```

Para definir el campo **descBook** de la tabla **authors**:

```
<tr>
<td bgcolor="#000000" class="filaTitulo">
<?php echo $html->tagErrorMsg('Book/descBook', '*') ?>Descripcion
</td>
<td bgcolor="#c0c0c0">
<?php echo $html->textarea('Book/descBook', array('rows' => '10'))?>
</td>
</tr>
```

Para definir el campo **author_id** de la tabla **authors**:

```
<tr>
<td bgcolor="#000000" class="filaTitulo">
<?php echo $html->tagErrorMsg('Book/author_id', '*') ?>Autor
</td>
<td bgcolor="#c0c0c0">
<?php echo $html->selectTag('Book/author_id', $authors, $selectedAuthor) ?>
</td>
</tr>
```

Para definir el campo **priceBook** de la tabla **authors**:

```
<tr>
<td bgcolor="#000000" class="filaTitulo">
```

* MÉTODOS SAVE Y DEL

Debemos saber que los métodos denominados **save** y **del** son propios de CakePHP y nos permiten almacenar nuevos registros o guardar cambios en los ya existentes (el primero) y eliminar una fila (el segundo método). Como es característico en este tipo de aplicaciones, no hay necesidad de definirlos explícitamente, de esta forma todo es más simple para el desarrollador.

```
<?php echo $html->tagErrorMsg('Book/priceBook', '*') ?>Precio
</td>
<td bgcolor="#c0c0c0">
<?php echo $html->input('Book/priceBook', array('size' => '40'))?>
</td>
</tr>
```

Para enviar el formulario, generamos un botón mediante el método **submit**, que recibe como argumento el propio texto:

```
<p><?php echo $html->submit('Guardar Cambios') ?></p>
```



Figura 19. La salida de las vistas se inserta dentro del template base del sitio, definido en el directorio app/views/layouts.

Otra vista disponible es la que se encuentra definida en **views/authors/index.thtml** y que permite mostrar el listado de autores disponibles. Primero generamos los títulos de la tabla, con las etiquetas correspondientes:

```
<tr>
<th align="left" colspan="2"><h1>Listado de autores</h1></th>
<th align="center" valign="middle"><?php echo $html->link("Agregar",
```

```

        "/authors/add"); ?></th>
</tr>
<tr bgcolor="#000000">
<th class="filaTitulo" align="left">Nombre</th>
<th class="filaTitulo" align="center" width="100">Editar</th>
<th class="filaTitulo" align="center" width="100">Eliminar</th>
</tr>

```

Y luego recorremos el array `$authors` para listar cada autor, y mostrar enlaces para acceder a editar y eliminar registros:

```

<?php if (count($authors)) { ?>
<?php foreach ($authors as $author): ?>
<tr bgcolor="#c0c0c0">
<td><?php echo $author['Author']['nameAuthor'].' '.$author
['Author']['lnameAuthor']; ?></td>
<td align="center"><?php echo $html->link('Editar', '/authors/edit/'.
$author['Author']['id']);?></td>
<td align="center"><?php echo $html->link('Eliminar', "/authors/delete/
{$author['Author']['id']}"), null, 'Desea eliminar este autor?'?></td>
</tr>
<?php endforeach; ?>
<?php } else { ?>
<tr bgcolor="#c0c0c0">
<td colspan="4">No se encontraron registros !</td>
</tr>
<?php } ?>

```

Fuente del template utilizado en el proyecto: www.templateyes.com.



RESUMEN

CakePHP es una popular alternativa a la hora de implementar soluciones utilizando la arquitectura Modelo Vista Controlador. En este capítulo, observamos de manera objetiva algunas de sus propiedades más interesantes a través de ejemplos prácticos, haciendo énfasis en las características particulares y la sintaxis de parte de sus funcionalidades.



ACTIVIDADES

PREGUNTAS TEÓRICAS

1 ¿Cuál cree que es la principal ventaja de trabajar con CakePHP?

2 ¿Utilizó Ruby On Rails? ¿Qué diferencias nota en referencia a CakePHP?

3 ¿Qué es el ActiveRecord?

4 Busque información acerca de ORM.

5 ¿En qué casos utilizaría la arquitectura MVC?

6 ¿Cuál es la función principal del modelo?

7 ¿Cuál es la función principal de los controladores?

8 ¿Cuál es la función principal de las vistas?

9 ¿Qué se almacena en el directorio app/views/layouts?

10 ¿Qué se almacena en el directorio app/webroot?

EJERCICIOS PRÁCTICOS

1 Modifique el ejemplo del muestrario de libros para que las rutas de las búsquedas sean de la forma /books/nameBook/nombreLibro.

2 Modifique el ejemplo del muestrario de libros para que al quitar un autor se eliminen también sus libros.

3 Implemente una relación de muchos a muchos en CakePHP.

Servicios web

Los servicios web son ya una parte importante de los desarrollos modernos. Por eso, en este capítulo daremos un recorrido por los componentes fundamentales que los constituyen y por las utilidades que ofrece PHP para generarlos y poder accederlos.

Historia	168
Capas	169
Protocolos	171
XML-RPC	171
SOAP	172
WSDL	174
UDDI	176
Servicios web en PHP	177
Extensiones disponibles	177
PEAR	181
Resumen	195
Actividades	196

HISTORIA

Un **servicio web** es, en resumidas cuentas, una aplicación que recibe peticiones de una aplicación y devuelve respuestas a otra. Para establecer la comunicación, se suelen utilizar protocolos estándares como **HTTP**. A su vez, los servicios web utilizan un conjunto de tecnologías aceptadas prácticamente por toda la industria de desarrolladores de software, siendo multiplataforma para su creación y consumo.

The screenshot shows a Wikipedia article page with the following details:

- Header:** Nuevas características · Registrarse/Entrar
- Page Title:** Web Services Protocol Stack
- Content Summary:** La Pila de protocolos para Servicios Web es una colección de protocolos para redes de Computadores que son utilizados para definir, localizar, implementar y hacer que un Servicio Web interactúe con otro. La Pila de Protocolos para servicios esta comprendida principalmente por cuatro áreas:
- List of Components:**
 - Servicio de Transporte: responsable del transporte de mensajes entre las Aplicaciones de red y los protocolos en los cuales se incluyen protocolos tales como HTTP, SMTP, FTP, así como también el más reciente Blocks Extensible Exchange Protocol (BEEP).
 - Mensajería XML: responsable por la codificación de mensajes en un formato común XML así que ellos puedan ser entendidos en cualquier extremo de una conexión de red. Actualmente, esta área incluye protocolos tales como XML-RPC, SOAP y REST.
 - Descripción del Servicio: usado para describir la interfaz pública de un Servicio Web específico. El formato de interfaz Web Services Description Language - WSDL es típicamente usado para este propósito.
 - Descubrimiento de servicios: centraliza servicios en un registro común tal que los servicios Web de la red puedan publicar su localización y descripción, y hace que sea fácil descubrir qué servicios están disponibles en la red. Actualmente, la API Universal Description Discovery and Integration - UDDI se utiliza normalmente para el descubrimiento de servicios.
- Text Below Summary:** La Pila de Protocolos para servicios también incluye un amplio rango de protocolos recientemente definidos: Business Process Execution Language - BPEL, SOAP Security Extensions: Digital Signature - SOAP-DSIG.
- Section:** [editar] Enlaces externos
- External Links:**
 - Lawrence Wilkes (updated Feb 2005) The Web Services Protocol Stack ↗
 - Alex Nghiêm (2003) The Basic Web Services Stack ↗
 - Ethan Cerami (2002) Top Ten FAQs for Web Services ↗
 - Pavel Kulchenko (2002) Web Services Acronyms, Demystified ↗
- Category:** Categoría: Protocolos de red
- Page Footer:** Esta página fue modificada por última vez el 14 dic 2008, a las 20:00.

Figura 1. Los servicios web han alcanzado gran popularidad en toda clase de proyectos, desde los simples hasta los complejos.

El formato de los mensajes que se intercambian está preestablecido, lo que permite que las aplicaciones en ambos lados de la comunicación puedan reconocerlos y responderlos generando otros nuevos.

XML es una de las opciones más utilizadas para dar marco a los mensajes. Una de las razones por las cuales se convirtió en el lenguaje elegido para trabajar con servicios web es que ofrece un formato de datos universal que permite adaptar o transformar fácilmente la información.

Pero más allá del lenguaje, una aplicación espera encontrar en un mensaje datos específicos, es decir, no sólo un XML bien formado. Para esto se utilizan diversas implementaciones de protocolos que son capaces de reconocer en un mensaje la parte correspondiente a la respuesta o a la petición hecha por la aplicación ubicada en el otro extremo de la comunicación.

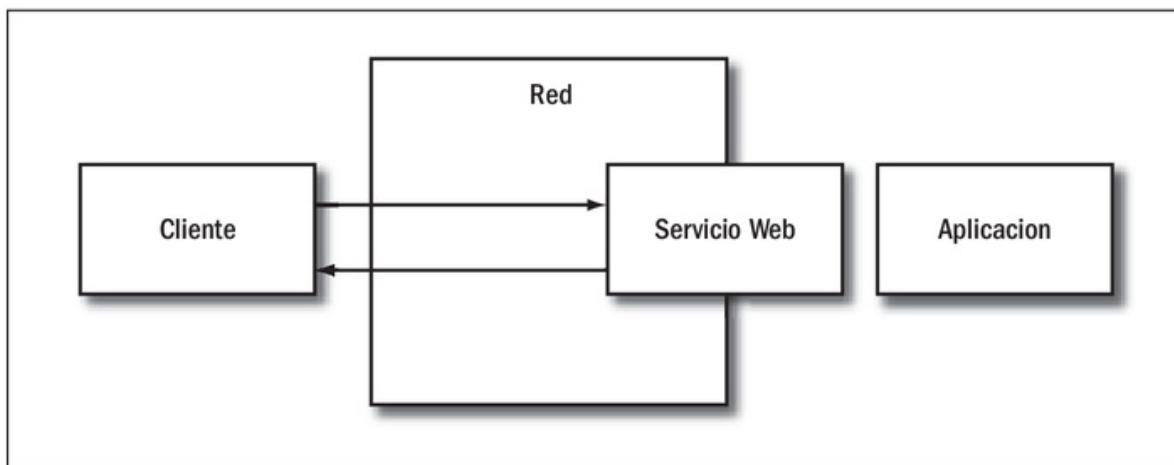


Figura 2. Un servicio web permite, entre otras cosas, neutralizar las posibles incompatibilidades entre aplicaciones.

Una de las ventajas de la utilización de servicios web por parte de las aplicaciones es la posibilidad de satisfacer pedidos de múltiples sistemas (quizás desarrollados en distintos lenguajes y ejecutándose bajo diferentes entornos) a partir de una misma fuente de datos. Esta clase de vinculación se da, incluso, como parte del proceso de desarrollo de muchas aplicaciones, añadiendo a ellas una capa extra (**interfaz**) que actúe a manera de servicio web. Otra utilidad consiste en el acceso a la funcionalidad de una aplicación desde cualquier máquina que posea una conexión a Internet, siempre y cuando el servicio esté disponible por este medio, claro. A la vez, mover las aplicaciones de escritorio y convertirlas en servicios web permite a los operadores no ligarse a un equipo específico para acceder, sino ampliar el espectro para utilizarlas desde distintos ámbitos.

CAPAS

Un servicio web se estructura en **capas**, cada una de las cuales tiene una funcionalidad específica, como por ejemplo, ubicar un determinado servicio, conocer las reglas para acceder a él e interactuar realizando peticiones y obteniendo respuestas. Cada una se implementa mediante el uso de protocolos abiertos, estandarizados en todo el mundo y, entre ellas, podemos citar las siguientes.

CAPA	DESCRIPCIÓN
Descubrimiento	Día a día, la oferta en cuanto a servicios web crece y, por ende, también aumenta la dificultad para encontrar uno que se adecue a lo que necesitamos. El proyecto UDDI (<i>Universal Description, Discovery, and Integration</i> – Descripción, Descubrimiento, e Integración Universales) pretende crear un directorio global de los servicios web disponibles y es hasta el momento el estándar más utilizado en esta capa.

CAPA	DESCRIPCIÓN
Descripción	Nos ofrece un mecanismo para describir la funcionalidad de un servicio y así poder interactuar desde nuestras aplicaciones. El estándar que mejor cumple este objetivo en estos momentos es WSDL (<i>Web Service Description Language - Lenguaje de Descripción de Servicios Web</i>).
Empaqueamiento	Para anular las incompatibilidades entre los distintos lenguajes de programación y sistemas operativos, se necesita aplicar un procesamiento extra a los mensajes. La estructura de las peticiones y las respuestas se define en esta capa, y el protocolo utilizado hoy en día es SOAP (<i>Simple Object Access Protocol - Protocolo Simple de Acceso a Objetos</i>), basado en XML. Veremos más acerca de este protocolo y de su vinculación con PHP en las próximas páginas.
Transporte	Una vez estandarizado el mensaje, debe ser enviado a destino a través de un protocolo preparado para tal fin. En estos momentos, el más popular es HTTP (<i>Hypertext Transport Protocol - Protocolo de Transporte de Hipertexto</i>), que es admitido por todos los navegadores y servidores web.
Red	Tanto el envío como la recepción de mensajes necesitan un medio físico para viajar de un punto a otro. El ámbito, una red, en la mayoría de los casos puede ser Internet (mediante el protocolo TCP/IP) o una subred.

Tabla 1. Capas que intervienen en el desempeño de un servicio web.




RFC1945 - Hypertext Transfer Protocol -- HTTP/1.0

[Internet RFC Index](#)
[Usenet FAQ Index](#)
[Other FAQs](#)
[Documents](#)
[Tools](#)

Search
[Search FAQs](#)
[Search RFCs](#)

IFC Home
[Cities](#)
[Countries](#)
[Hospitals](#)
[Web Hosting Ratings](#)

Search the RFC Archives

Google™ Custom Search

Or Display the document by number

[[RFC Index](#) | [Usenet FAQs](#) | [Web FAQs](#) | [Documents](#) | [Cities](#) | [Patents](#) | [Abstracts](#)]

Network Working Group
Request for Comments: 1945
Category: Informational

T. Berners-Lee
MIT/LCS
R. Fielding
UC Irvine
H. Frystyk
MIT/LCS
May 1996

Figura 3. El protocolo HTTP conserva la popularidad adquirida en los comienzos mismos de Internet.

PROTOCOLOS

A continuación, haremos un recorrido por los protocolos y lenguajes más utilizados en el desarrollo de servicios web: XML-RPC, SOAP, WSDL, y UDDI.

XML-RPC

XML-RPC (*XML Remote Procedure Call*, Llamadas a Procedimientos Remotos por XML) se utiliza para establecer conexiones entre sistemas remotos y se convirtió en el primer protocolo realmente popular basado en XML. Fue desarrollado por la empresa **Userland Software** (www.userland.com) y, hasta el día de hoy, muchas aplicaciones lo incorporan como una opción válida.



Figura 4. La empresa Userland Software pone a disposición de los usuarios servicios web para realizar pruebas.

XML-RPC es una implementación basada en el modelo **RPC**, utilizado para establecer conexiones y facilitar transacciones entre dos sistemas remotos. Entre los sucesores

III SEGURIDAD EN TRANSACCIONES

Debemos tener en cuenta que el proyecto denominado *Global XML Web Services* está liderado, entre otras empresas, por Microsoft y trata temas referidos a la interoperabilidad entre los distintos servicios web. Se encuentra orientado a la seguridad en lo que hace al intercambio de mensajes entre ellos, un tema de suma importancia.

más conocidos, se encuentran **DCOM** y **CORBA**. Al ser un protocolo que permite realizar llamadas a funciones ubicadas en un servidor a través de HTTP, su utilización es intuitiva y permite obtener resultados concretos a partir de pocas líneas de código.



Figura 5. XML-RPC revalorizó la utilización del metalenguaje XML como formato de intercambio de datos entre aplicaciones.

En el sitio www.xmlrpc.com, podemos encontrar una gran cantidad de información y recursos que incluyen servicios web listos para utilizar (www.xmlrpc.com/directory/1568/services).

SOAP

SOAP es el protocolo de comunicación que más utilizan los servicios web para interactuar entre sí. Deriva del protocolo RPC e intenta mantener dos de sus características: la simplicidad y la potencia.

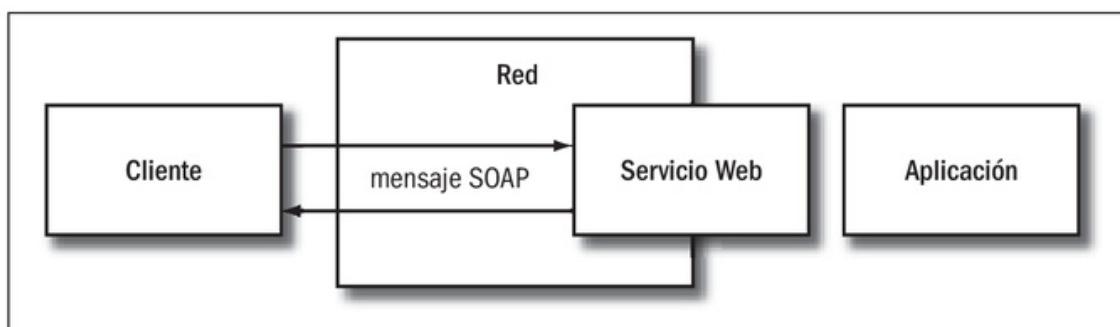


Figura 6. SOAP se ha convertido en un estándar en lo referido al envío y recepción de mensajes dentro del ámbito de los servicios web.

Puesto que su fin es establecer una forma de comunicación entre aplicaciones diversas tanto en características técnicas como en finalidad, SOAP surgió como resultado de un consenso entre un grupo compuesto por importantes empresas, como por ejemplo, UserLand, Ariba, Commerce One, Compaq, Developmentor, HP, IBM, IONA, Lotus, Microsoft y SAP, quienes lo propusieron a la W3C para que lo incluyera entre sus estándares. Una vez sucedido esto, las distintas herramientas de desarrollo (PHP por caso) tomaron la especificación oficial (disponible en www.w3.org/TR/soap12-part1) para incluir sus propias implementaciones. Al plantear los principios sobre los cuales se iba a comenzar el desarrollo de este protocolo, se pensó básicamente en dos características: que fuera simple y que fuera escalable para poder ampliarlo a través del tiempo.

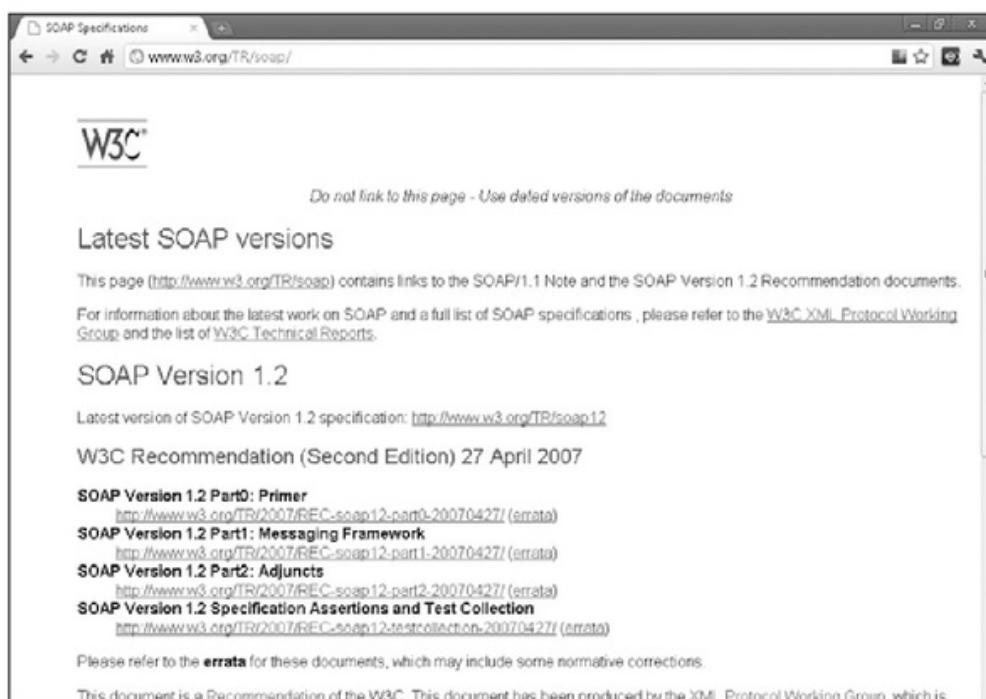


Figura 7. SOAP cuenta con el apoyo de importantes empresas y dispone de una especificación oficial publicada en el sitio de la W3C.

Como mencionamos anteriormente, la finalidad de un protocolo como SOAP es la de proveer un mecanismo estándar para empaquetar mensajes basado en XML (vimos más



EL MEDIO DE COMUNICACIÓN

Es importante saber que la acción fundamental de un servicio web es comunicar aplicaciones, por lo que, la parte visual es nula o inexistente: tanto los mensajes de petición como los de respuesta incluyen sólo los datos considerados como elementales. Las aplicaciones que hacen uso de ellos deberán ocuparse de proporcionar la interfaz para mostrarlos.

información acerca de este metalenguaje en el **Capítulo 3**). Un mensaje SOAP consta de varios elementos, cada uno bien definido en la especificación oficial del protocolo:

ELEMENTO	DESCRIPCIÓN
Envelope	Es el elemento raíz del documento XML generado y sirve para identificar, ante los eventuales receptores, que se trata de un mensaje SOAP. Además, entre otras cuestiones, podemos identificar la versión de SOAP que estamos utilizando.
Header	Este elemento se ubica dentro de Envelope. Es opcional y, en caso de existir, debe estar antes del elemento Body. Aquí se incluye información de control acerca del mensaje, como por ejemplo el destinatario y un conjunto de opciones de entrega.
Body	En esta parte del documento XML, se ubican, además de otros elementos específicos, los datos que formarán parte del mensaje SOAP. Su inclusión es obligatoria.
Fault	Se ubica dentro del elemento Body y permite representar que han ocurrido errores o fallos al procesar una petición SOAP. Es opcional. Admite cuatro subelementos: faultcode (código asignado al error), faultstring (descripción del error), faultactor (información acerca de quién causó el error) y detail (detalles adicionales acerca del error que se ha producido, para que el cliente pueda interpretarlo de mejor manera. Admite subelementos).

Tabla 2. Elementos de un documento SOAP.

La cabecera mantiene información de control acerca del origen del mensaje y el receptor, mientras que el cuerpo contiene el mensaje en sí, es decir, la información que se desea enviar (una petición/pregunta o una contestación/respuesta).

WSDL

Un documento **WSDL** presenta las operaciones disponibles del servicio web al que hace referencia. Se trata de un documento XML que especifica qué datos deberá contener un mensaje de solicitud (petición) y cómo será un mensaje de respuesta. Dentro del documento, se define, además, dónde está disponible el servicio y qué protocolo de comunicaciones utiliza. WSDL describe un esquema para los mensajes que se intercambiarán. En definitiva, un documento WSDL contiene la información necesaria para generar mensajes SOAP. Tal como sucede con

III LA OPCIÓN A WSDL

Una alternativa a WSDL está dada por los **XML Schemas** (www.w3.org/XML/Schema), que permiten definir a través de reglas la estructura de un documento XML y admiten que las aplicaciones validen su estructura. Como mencionamos, las capas relacionadas con los servicios web no están ligadas a herramientas particulares, algo que da versatilidad a los desarrollos.

los documentos XML tradicionales, la interpretación de un archivo WSDL por parte de un usuario no es difícil y nos permite entender de manera clara cómo una aplicación interacciona con el servicio.

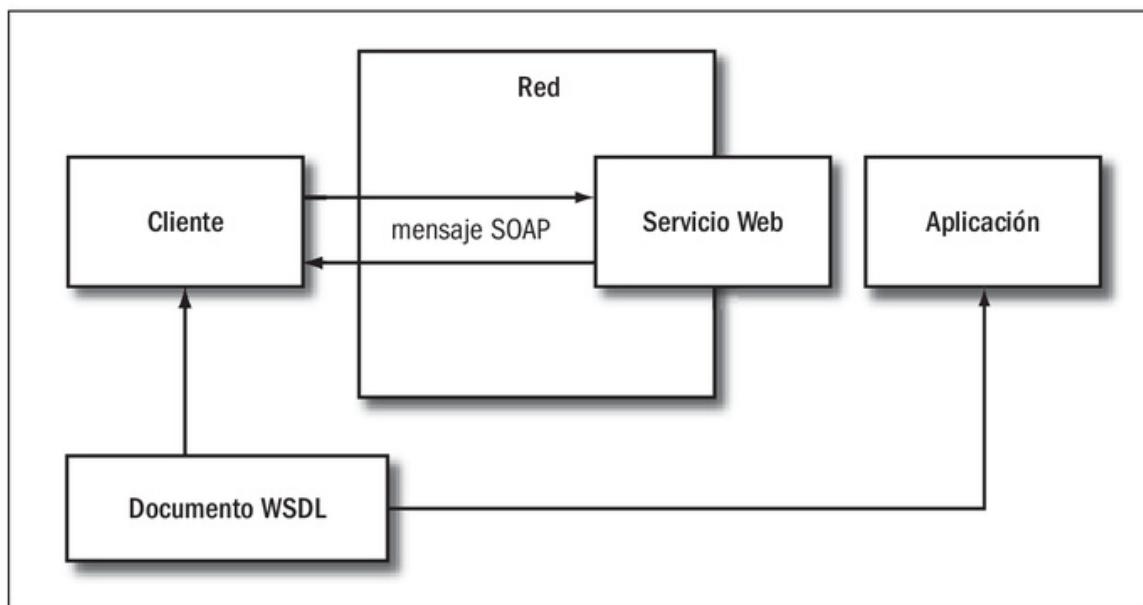


Figura 8. WSDL permite a las aplicaciones conectarse entre sí para luego poder intercambiar informaciones.

Uno de los elementos más importantes de un documento WSDL es **definitions**, y es allí donde se ubican las claves que nos permitirán interactuar con un servicio. Dentro de **definitions**, podemos encontrar los siguientes elementos:

- **portType**: detalla las operaciones ofrecidas por el servicio, que podrán luego ser accedidas por los clientes. Hay cuatro tipos de operaciones disponibles, entre las que se encuentran **One-way** (puede recibir mensajes, pero no puede dar una respuesta), **Request-response** (puede recibir peticiones y dar una respuesta), **Solicit-response** (puede enviar una petición y esperar por una respuesta) y **Notification** (puede enviar una petición, pero no puede esperar por una respuesta).
- **message**: da una definición de cada uno de los mensajes utilizados por el servicio para reconocer de manera directa sus características.

III SITIOS WEB

En la actualidad, los servicios web son tomados por las empresas que brindan sus servicios a través de Internet como una alternativa más para hacer llegar sus prestaciones a quienes quieran hacer uso de ellas, ganando versatilidad en lo referido a la administración de datos. De esta forma proporcionan una mayor versatilidad a los usuarios.

- **types:** detalla los tipos de datos utilizados por el servicio.
- **binding:** declara los protocolos de comunicación utilizados.
- **port:** sirve para definir al último receptor del mensaje de petición mediante un enlace (una URL) y una dirección de red.

W3C Note

W3C

Web Services Description Language (WSDL) 1.1

W3C Note 15 March 2001

This version:
<http://www.w3.org/TR/2001/NOTE-wsdl-20010315>

Latest version:
<http://www.w3.org/TR/wsdl>

Authors (alphabetically):
Erik Christensen, Microsoft
Francisco Curbera, IBM Research
Greg Meredith, Microsoft
Sanjiva Weerawarana, IBM Research

Copyright© 2001 Akiba, International Business Machines Corporation, Microsoft

Abstract

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services). WSDL is extensible to allow description of endpoints and their messages regardless of what message formats or network protocols are used to communicate; however, the only bindings described in this document describe how to use WSDL in conjunction with SOAP 1.1, HTTP GET/POST, and MIME.

Status

Figura 9. Un documento WSDL facilita la interacción con servicios web mediante la automatización de áreas frecuentes, la especificación de WSDL se encuentra homologada por el W3C.

Como veremos en las próximas páginas, es común que las aplicaciones que brindan su funcionalidad a través de un servicio web pongan a disposición de los que quieran utilizarlo un documento WSDL para simplificar su utilización. Luego, desde distintos lenguajes de programación, será posible, a través de este archivo, realizar peticiones y recibir la respuesta correspondiente desde cualquier plataforma.

En el sitio oficial de la W3C, se mantiene la trascipción del detalle de la especificación, que podemos consultar en www.w3.org/TR/wsdl. Allí encontraremos, entre otras cosas, ejemplos comentados por los propios autores: www.w3.org/2001/03/14-annotated-WSDL-examples.

UDDI

UDDI es un protocolo que nos permite la publicación y localización de servicios web. Al ser tan imponente la oferta y teniendo en cuenta que las perspectivas siguen yendo en el mismo sentido, ubicar un determinado servicio es de sumo interés.

Figura 10. UDDI plantea la importancia de catalogar servicios web para permitir a los usuarios encontrar una solución acorde con sus necesidades.

El registro se realiza a través de un documento XML que se pone a disposición del público en un catálogo global. Allí se ingresan datos relacionados con el responsable de la aplicación, el tipo de servicio brindado, su descripción y, eventualmente, un archivo WSDL. El servicio puede ser público o restringido para usuarios habilitados.

Podemos obtener más información acerca de las características ofrecidas por el directorio UDDI en el sitio oficial: <http://uddi.xml.org>.

SERVICIOS WEB EN PHP

PHP pone a nuestra disposición un conjunto de extensiones para generar y acceder servicios web. Se incluyen funciones para SOAP, WDDX y XML-RPC. Además, veremos las opciones ofrecidas por PEAR para generación y acceso a servicios.

Extensiones disponibles

En la distribución estándar del lenguaje, contamos con la extensión **SOAP**, que nos permite acceder y crear servicios web. Para habilitarla en sistemas Linux, debemos configurar PHP con la opción **soap**, y, en Windows, descomentar la línea correspondiente al uso de este protocolo, ubicada en el archivo **php.ini**:

```
extension=php_soap.dll
```

Para la versión 5 de PHP, la extensión se reescribió completamente (está desarrollada en lenguaje C, mientras que la anterior fue escrita en PHP), y más allá de cuestiones muy puntuales, la mayor diferencia se nota en la velocidad de acceso a los servicios, sustancialmente mejorada. Hay dos clases principales: **SoapClient** para acceder a servicios y **SoapServer** para brindarlos.

Si disponemos de la ruta al documento **wsdl** que describe el servicio, podemos indicarlo como argumento al constructor **SoapClient** y comenzar a utilizar las funciones disponibles en la aplicación automáticamente:

```
<?php

$wsdl = "http://www.direccion.com/archivo.wsdl";
$cliente = new SoapClient($wsdl);
$cliente->nombrefuncion("valorParametro");

?>
```

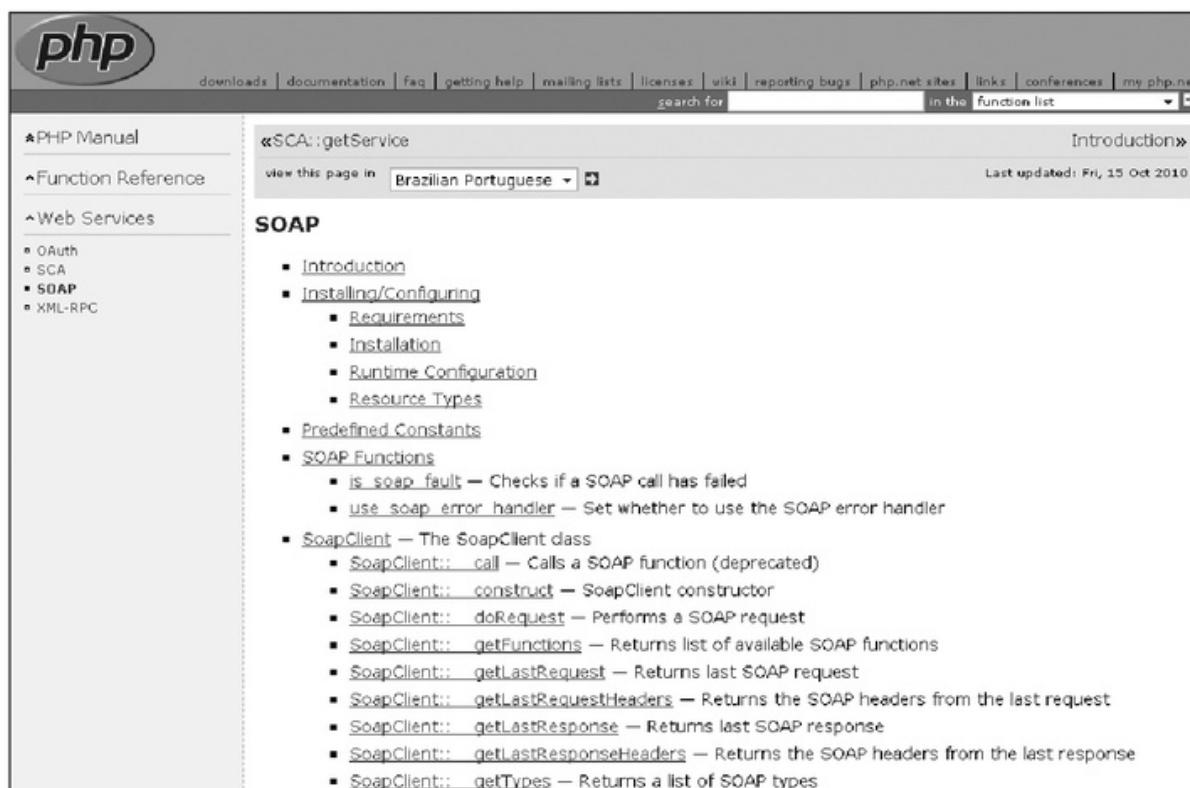


Figura 11. La extensión **SOAP** de PHP viene incluida en la distribución oficial del lenguaje.

Para invocar funciones alternativamente podemos utilizar el método `__soapCall`, que recibe el nombre de la función más un array con los argumentos:

```
<?php

$wsdl = "http://www.direccion.com/archivo.wsdl";
$cliente = new SoapClient($wsdl);
$cliente->__soapCall("nombreFuncion", array("valor1", "valor2"));

?>
```

Tanto las funciones disponibles como sus argumentos podemos observarlas en el archivo `wsdl`, o bien, mediante el método `__getFunctions`:

```
<?php

$wsdl = "http://www.direccion.com/archivo.wsdl";
$cliente = new SoapClient($wsdl);
print_r($cliente->__getFunctions());

?>
```

En el archivo `php.ini`, contamos con algunas directivas de configuración relacionadas que pueden ser útiles cuando conocemos la ruta al `wsdl`. Están ubicadas en la sección `soap` y son las siguientes:

```
[soap]
soap.wsdl_cache_enabled=1
soap.wsdl_cache_dir="/tmp"
```

III COMPROBAR LA INTEGRIDAD DE LOS MENSAJES

Entre las alternativas para realizar la comprobación de la integridad de los mensajes están **WS-Security** (cifrado, firmas digitales y encriptación), **WS-License** (autorización del acceso a un servicio), **WS-Routing** (enrutamiento de mensajes) y **WS-Referral** (configuración dinámica) son algunas de las especificaciones relacionadas con la seguridad en servicios web.

```
soap.wsdl_cache_ttl=86400
```

La primera directiva indica si el documento **wsdl** dado como argumento a **SoapClient** se guardará en caché. La segunda, el directorio en el cual se almacenará. Y la tercera, el tiempo de vida entre una actualización y otra. Todas estas directivas están disponibles en PHP versión 5 y superiores.

Si, por el contrario, el servicio al que queremos acceder no cuenta con un archivo de definición, deberemos incluir los parámetros manualmente tanto en el constructor como a los métodos invocados (mediante el método **_call** que recibe el nombre y un array con los parámetros particulares):

```
<?php

$args[“location”] = “http://www.direccion.com/”;
$args[“uri”] = “pre:namespaceName”;

$cliente = new SoapClient(NULL, $args);

$param[] = new SoapParam(“valorParametro”, “nombreParametro”);

unset($args);
$args[“uri”] = “pre:namespaceName”;
$args[“soapaction”] = “pre:namespaceName#nombreFuncion”;

echo $cliente->_call(“nombreFuncion”, $param, $args);

?>
```

Encontraremos más información acerca de la extensión **SOAP** en el manual de PHP, ubicado en el sitio oficial: www.php.net.

III ALTERNATIVA A LA EXTENSIÓN SOAP

La extensión SOAP provista por PHP no es la única opción que nos ofrece el lenguaje para trabajar con este protocolo. **NuSOAP** es una alternativa que viene en la distribución de PHP y que, además, incorpora soporte para WSDL. Está escrita en PHP, y podemos obtener más información en <http://sourceforge.net/projects/nusoap>.

PEAR

El repositorio oficial de clases de PHP, PEAR (veremos más información acerca de él en el **Capítulo 8**), nos brinda varias opciones para generar y hacer uso de servicios web.

Como mencionamos anteriormente, XML-RPC nos permite transportar datos codificados en XML entre dos máquinas, utilizando HTTP, y PEAR incorpora una implementación para hacer uso del protocolo. Para instalar el paquete **XML-RPC** en nuestro sistema, debemos ejecutar lo siguiente desde la línea de comandos:

```
c:\> pear install XML-RPC
```

Tanto las llamadas como las respuestas son generadas en XML, por lo que, para realizar una petición, deberemos generar en nuestras aplicaciones un documento de este tipo. El paquete **XML-RPC** automatiza esta tarea a partir de la clase **XML_RPC_Message** (para crear un mensaje) y la función **XML_RPC_decode** (para decodificar la respuesta). En el siguiente ejemplo, utilizamos un servicio web puesto a disposición del público por Userland Software que, dado un código numérico, nos devuelve el estado correspondiente. Primero, incluimos el archivo **RPC.php**:

```
require_once 'XML/RPC.php';
```

El constructor **XML_RPC_Client** recibe como parámetros:

- Una ruta interna para ubicar la aplicación detrás del servicio web (URI).
- Servidor remoto.
- Puerto (opcional, por defecto 80).
- Proxy (opcional).
- Puerto del proxy (opcional).
- Nombre de usuario del usuario del proxy (opcional).
- Contraseña del usuario del proxy (opcional).



EJEMPLO EN LÍNEA

En el artículo online que se encuentra en la dirección que comentamos a continuación, podemos encontrar un buen ejemplo de uso de Web Services desde PHP, utilizando PEAR, con el código fuente incluido y la utilización de un documento WSDL para descripción del servicio, en la dirección www.onlamp.com/pub/a/php/2003/07/03/php_amazon_soap.html.

```
$cliente = new XML_RPC_Client('/RPC2', 'betty.userland.com');
```

The screenshot shows the PEAR package page for XML_RPC. At the top, there's a navigation bar with links for Main, Support, Documentation, Packages, Package Proposals, Developers, and Bugs. Below that, a search bar says "Search for" and "in the Packages". A "Register / Login" button is on the right. The main content area has a header "Top Level :: Web Services" and "Package Information: XML_RPC". Below that is a menu with "Main", "Download", "Documentation", "Bugs", and "Trackbacks". A link "Show All Changelogs" is present. The "Version" section shows "1.5.4" with an "Information" link. Under "Information", there's an "Easy Install" link ("pear install XML_RPC-1.5.4") and a "Download" link ("1.5.4"). Below this, "Release date: 2010-07-03 10:19 UTC" and "Release state: Stable" are listed. A "Changelog" section contains a list of changes: "Change ereg functions to preg due to deprecation (Request 17546 and then some.)", "Fix bugs in XML_RPC_Dump error detection process.", and "Escape XML special characters in key names of struct elements. Bug 17360.". A "Dependencies" section lists requirements: "PHP Version: PHP 4.2.0 or newer" and "PHP Extension: xml". At the bottom, two previous versions are listed: "1.5.3" (stable, 2010-01-14) and "1.5.2" (stable, 2009-08-18).

Figura 12. PEAR mantiene entre sus múltiples opciones la de acceder y generar servicios web, utilizando el protocolo XML-RPC.

Con **XML_RPC_Value**, definimos los valores que se enviarán como argumento al servicio (valor y tipo de dato) como muestra la siguiente linea:

```
$args[] = new XML_RPC_Value(3, 'int');
```

A través de **XML_RPC_Message**, generamos el mensaje. Recibe como argumentos el nombre de la función remota y sus argumentos:

```
$mensaje = new XML_RPC_Message('examples.getStateName', $args);
```

III DOCUMENTO WSDL

Es importante recordar que un documento WSDL puede contener información referida a más de un servicio web. Al tratarse de un documento XML, la estructuración de los datos está bien definida y no deja lugar a dudas, de esta forma permite a las aplicaciones cliente encontrar exactamente la información que necesitan.

Con **send**, enviamos el mensaje generado:

```
$respuesta = $cliente->send($mensaje);
```

Los métodos **faultCode** y **faultString** permiten saber más acerca de un error cometido:

```
if ($respuesta->faultCode()) {
    echo "Ha ocurrido un error: ".$respuesta->faultString();
    exit;
}
```

A partir de la respuesta (**value**), **XML_RPC_decode** realiza una decodificación, devolviendo el valor, de la siguiente manera:

```
$estado = XML_RPC_decode($respuesta->value());
echo "La respuesta devuelta es $estado";
```

Lo anterior nos da como salida el siguiente código:

```
La respuesta devuelta es Arizona
```

Lo que espera recibir el servicio es algo como lo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
    <methodName>examples.getStateName</methodName>
    <params>
        <param>
            <value><int>3</int></value>
        </param>
    </params>
</methodCall>
```

Y la respuesta, en caso de ser positiva, es:

```
<methodResponse>
  <params>
    <param>
      <value><string>Arizona</string></value>
    </param>
  </params>
</methodResponse>
```

The screenshot shows the PEAR package information page for XML_RPC2. At the top, there's a navigation bar with links for Main, Support, Documentation, Packages, Package Proposals, Developers, and Bugs. Below the navigation, it says "List Packages | Search Packages | Statistics | Channels". The main title is "Package Information: XML_RPC2". Underneath, there are tabs for Main, Download, Documentation, Bugs, and Trackbacks. The "Main" tab is selected. On the left, there are sections for "Summary" (describing it as a client/server library), "Current Release" (version 1.0.0 released on 2010-10-10), "Description" (explaining its purpose as a simple remote procedure call protocol), "Maintainers" (listing Sergio Carvalho, Fabien MARTY, and Alan Langford), and "More Information" (links to source tree, RSS feed, and download statistics). On the right, there are sections for "License" (LGPL) and "Bug Summary" (listing 7 open bugs, an average age of 358 days, and an oldest bug from 1237 days ago).

Figura 13. El paquete XML-RPC de PEAR nos permite transformar la respuesta obtenida a un array PHP.

PEAR provee paquetes especialmente diseñados para acceder a servicios web particulares, entre los que podemos citar a:

Services_Amazon
Services_Blogging
Services_Compete

III DESCRIPCIÓN DE SERVICIOS CON WSDL

Debemos saber que WSDL se encarga de describir un servicio web una vez que éste fue publicado. Esta información tendrá un gran valor para las aplicaciones cliente que quieran acceder al servicio, ya que por ese medio podrán conocer las funciones disponibles y aprovecharlas sin necesidad de cometer errores en su manejo.

Services_Delicious
Services_Digg
Services_DynDNS
Services_ExchangeRates
Services_Google
Services_Hatena
Services_OpenSearch
Services_Pingback
Services_SharedBook
Services_Technorati
Services_TinyURL
Services_Trackback
Services_W3C_HTMLValidator
Services_Weather
Services_Webservice
Services_Yadis
Services_Yahoo
Services_YouTube

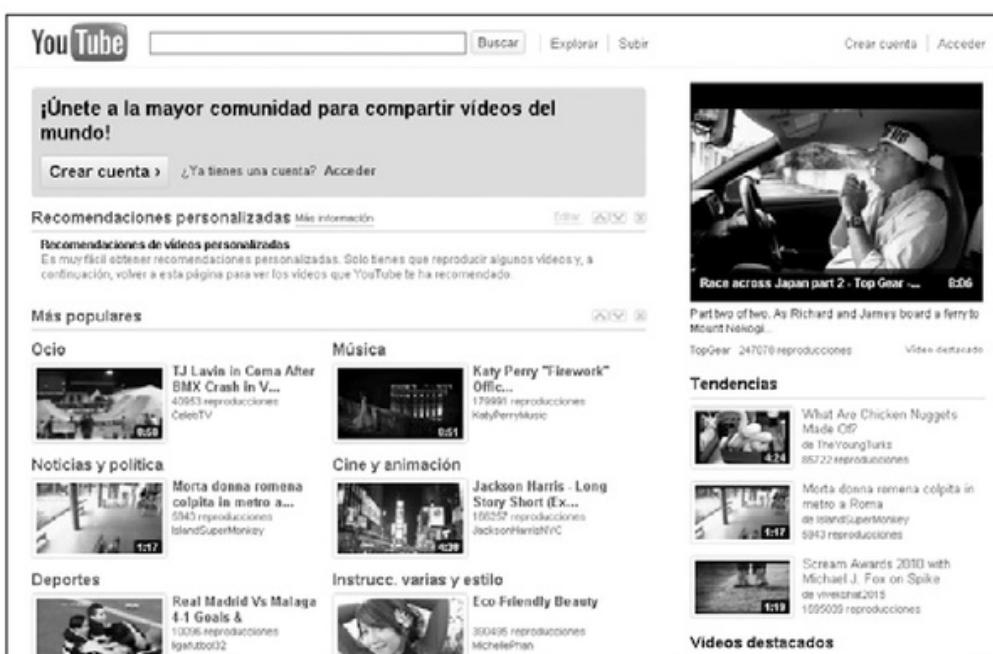


Figura 14. YouTube es uno de los sitios que mantienen un paquete específico para que los desarrolladores puedan acceder a través de PEAR.

Muchos de estos servicios requieren un registro previo, casi siempre gratuito, para ser utilizados por los desarrolladores. La forma de uso no es para nada compleja, y en pocos minutos podemos obtener aplicaciones funcionales. Tomemos por ejemplo el servicio web de la W3C, que nos permite saber si un documento contiene sólo HTML válido o no:

```
pear install -a Services_W3C_HTMLValidator
```

Podemos darnos cuenque que en el siguiente trozo de código, nos encargamos de realizar la validación del HTML de distintos sitios web, entre ellos Google, Clarín y W3C, veamos la forma adecuada de implementar esta función:

```
<?php

require_once 'Services/W3C/HTMLValidator.php';

$w3c = new Services_W3C_HTMLValidator();

$html[] = 'http://www.w3c.org/';
$html[] = 'http://www.google.com.br/';
$html[] = 'http://www.clarin.com/';
foreach ($html as $v) {
    $respuesta = $w3c->validate($v);

    if (!$respuesta->isValid()) {
        echo '<li>El documento '.$v.' no es valido.';
    } else {
        echo '<li>El documento '.$v.' es valido.';
    }
}
/*
El documento http://www.w3c.org/ es valido.
El documento http://www.google.com.br/ no es valido.
El documento http://www.clarin.com/ no es valido.
*/
?>
```



DELICIOUS

Delicious (www.delicious.com) es un sitio que nos permite compartir nuestros marcadores o bookmarks con otros usuarios y almacenarlos en linea. Es importante que tengamos en cuenta que PEAR nos ofrece una interesante alternativa desarrollada para acceder a algunos de los servicios brindados por esta muy popular herramienta.

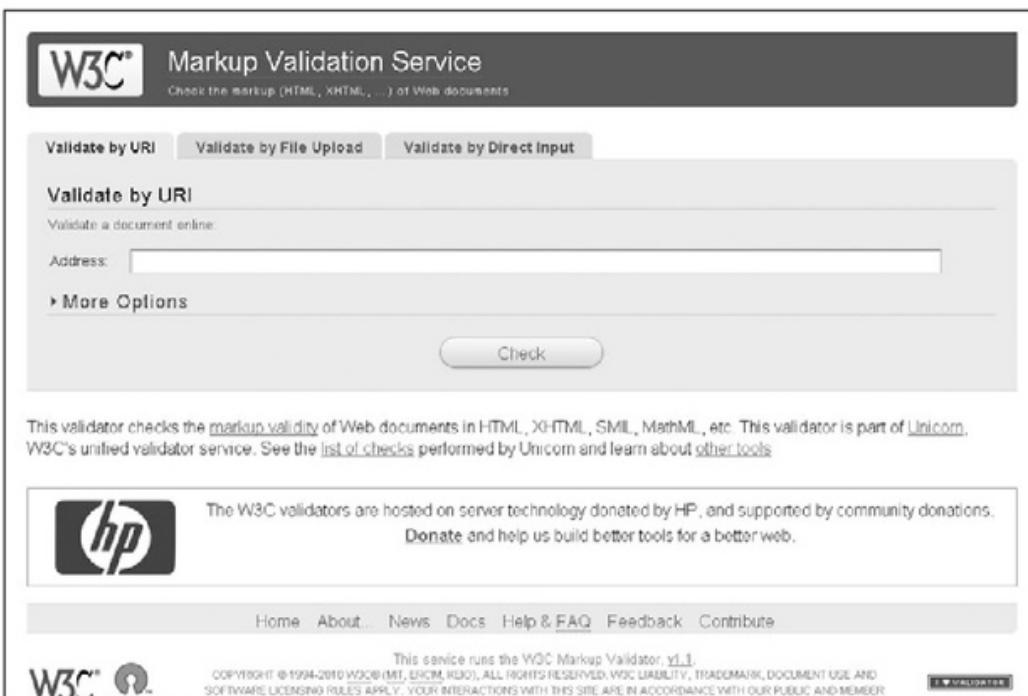


Figura 15. La W3C pone a disposición de los usuarios un servicio web para validar que el código de las páginas se apegue a los estándares oficiales.

Desde hace algún tiempo, **Yahoo** viene ofreciendo más y más herramientas a los desarrolladores, entre las que se cuenta la posibilidad de acceder, mediante servicios web, a las características de los sitios pertenecientes a la compañía. En el siguiente ejemplo, utilizaremos el API de **Flickr** (www.flickr.com), un website que se dedica al almacenamiento de imágenes subidas por los usuarios.

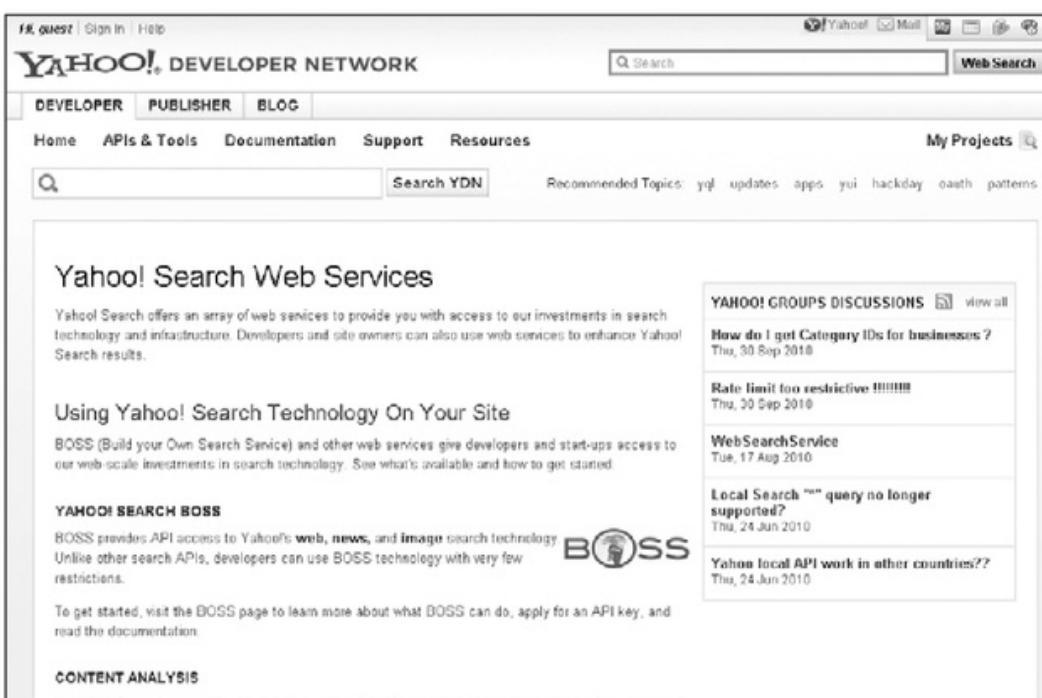


Figura 16. Yahoo es una de las empresas que más importancia da al desarrollo de servicios web, poniendo múltiples herramientas a disposición de los programadores.

Primero deberemos registrarnos en el sitio (para eso necesitaremos una cuenta de e-mail de Yahoo) y así poder obtener una clave de usuario. Una vez obtenida, es posible acceder a los servicios.

Entre otras muchas variantes, Flickr permite acceder a sus aplicaciones a través de **REST** (*Representational State Transfer*). Trabaja, al igual que XML-RPC y SOAP, valiéndose del protocolo HTTP, pero su forma de acceso es un poco más directa ya que incluye la ruta y los argumentos al servicio dentro de la petición URL (**query string**), por ejemplo:

```
http://www.nombre-sitio.com/nombre-metodo/?arg1=a&arg2=b
```

A diferencia de lo que ocurre con SOAP, los argumentos pasados al servicio son siempre de tipo **string**.



Figura 17. Flickr es un sitio que se dedica a la publicación de imágenes y de fotografías subidas por sus propios autores.

III ALTERNATIVA A LOS PROTOCOLOS OFICIALES

El término **REST** describe aquellos servicios web que han sido desarrollados utilizando protocolos no propietarios. Generalmente, están basados en XML y surgieron a partir de las necesidades no satisfechas por los protocolos oficiales. El acceso por este medio se caracteriza por ser sencillo y libre de complicaciones, por eso se convierte en una excelente alternativa.

Flickr ofrece muchísimas alternativas, como por ejemplo, listar imágenes de un determinado usuario, obtener los últimos ingresos, agregar fotos, etcétera. En el siguiente proyecto, realizaremos una búsqueda de imágenes a partir de un término dado:

```
http://api.flickr.com/services/rest/?method=flickr.photos.search&tags=ice&per_page=10&api_key=xxxxxxxxxxxxxx
```

En donde:

method: identifica el método al que queremos invocar.

tags: permite incluir etiquetas para refinar la búsqueda, separadas por comas.

per_page: define la cantidad de resultados por página (**page** especifica la página recuperada).

api_key: es la clave de identificación obtenida desde Flickr.

La respuesta es un documento XML como el siguiente:

```
<rsp stat="ok">
  <photos page="1" pages="296" perpage="10" total="2959">
    <photo id="5097554736" owner="52298200@N03" secret="64ce6f6532" server="4125" farm="5" title="Beatrix" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5084213521" owner="51387784@N02" secret="419db840a3" server="4012" farm="5" title="" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5081582746" owner="23585297@N00" secret="636302f20e" server="4043" farm="5" title="Nora as The Bride from Kill Bill" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5093469155" owner="52534620@N04" secret="beff7aeb53" server="4147" farm="5" title="Fierce eyes" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5093469143" owner="52534620@N04" secret="26a412f67" server="4153" farm="5" title="Kill Bill?" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5093469191" owner="52534620@N04" secret="fa03d10157" server="4131" farm="5" title="Parry" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5093469150" owner="52534620@N04" secret="6fc3b6e5e2" server="4084" farm="5" title="Dazzle" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5093469181" owner="52534620@N04" secret="0024d88574" server="4103" farm="5" title="Ready to Strike!" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5082660500" owner="44653010@N08" secret="725320915b" server="4089" farm="5" title="Kill Bill RT" ispublic="1" isfriend="0" isfamily="0" />
    <photo id="5076976874" owner="38109181@N05" secret="0aed315d2c" server="4001" farm="5" title="IM&GO157" ispublic="1" isfriend="0" isfamily="0" />
  </photos>
</rsp>
```

Figura 18. La utilización de REST nos permite acceder fácilmente a un servicio web, y obtener las respuestas a partir de peticiones por URL, siempre utilizando XML como formato preferido para transferir datos.

III ESTABLECER CONTACTO ENTRE APLICACIONES

Es importante mencionar que los servicios web surgieron a partir de la necesidad de que sistemas heterogéneos pudieran mantenerse conectados intercambiando la información necesaria. El auge de Internet aceleró más y más esta búsqueda, la cual fue impulsada por las grandes empresas del sector informático.

Debemos saber que si utilizamos el paquete denominado **HTTP_Request** presente en PEAR, podemos hacer más elegante el proceso de petición mediante el encapsulado los parámetros adecuados. Para utilizarlo, primero será necesario incluir el archivo **Request.php**, con la línea:

```
require_once 'HTTP/Request2.php';
```

Luego de ello debemos indicar la dirección URL en la cual se va a resolver el requerimiento que hemos indicado:

```
$peticion = new HTTP_Request2('http://api.flickr.com/services/rest/',
HTTP_Request2::METHOD_GET);
```

Es importante destacar que mediante el método denominado **setQueryVariable** incluimos los argumentos al servicio web:

```
$url = $peticion->getUrl();

$url->setQueryVariable('api_key', 'xxxxxxxxxxxxxxxxxxxx');
$url->setQueryVariable('method', 'flickr.photos.search');
$url->setQueryVariable('tags', 'kill bill');
$url->setQueryVariable('per_page', '10');
```

A través del método **send** enviamos la petición generada:

```
$respuesta = $peticion->send();
```

Y, si todo fue bien, recuperamos la respuesta, en este caso un documento XML:



XML

XML se presenta como una alternativa excelente en la interacción de bases de datos y aplicaciones o portales desarrollados en Flash con Actionscript. PHP también es otro lenguaje que maneja XML como puente intermedio entre Flash y este lenguaje, lo que permite el desarrollo simple de sitios y portales con e-commerce entre otros.

```

if (200 == $respuesta->getStatus()) {
    echo $respuesta->getBody();
}

```

Para manejar en detalle el fragmento devuelto, contamos con el paquete denominado **XML_Unserializer**, que, básicamente, se encarga de traducir un XML a estructuras PHP (arrays y objetos de datos, por ejemplo). Este paquete se incluye en forma predeterminada al instalar el paquete llamado **XML_Serializer**, de la manera que vemos en el siguiente código:

```
pear install -a XML_Serializer
```

Para continuar, es importante recordar que primero debemos preocuparnos de incluir el archivo denominado **Unserializer.php** y, posteriormente, procedemos a realizar la creación de un objeto **XML_Unserializer**:

```

require_once 'XML/Unserializer.php';
$xmlu = new XML_Unserializer();

```

Con **setOption**, podemos definir algunas opciones adicionales. Por ejemplo, será posible acceder a características tales como incluir valores de atributos, guardar la salida en un array, y también utilizar el juego de caracteres ISO-8859-1, para lo cual usamos el siguiente trozo de código:

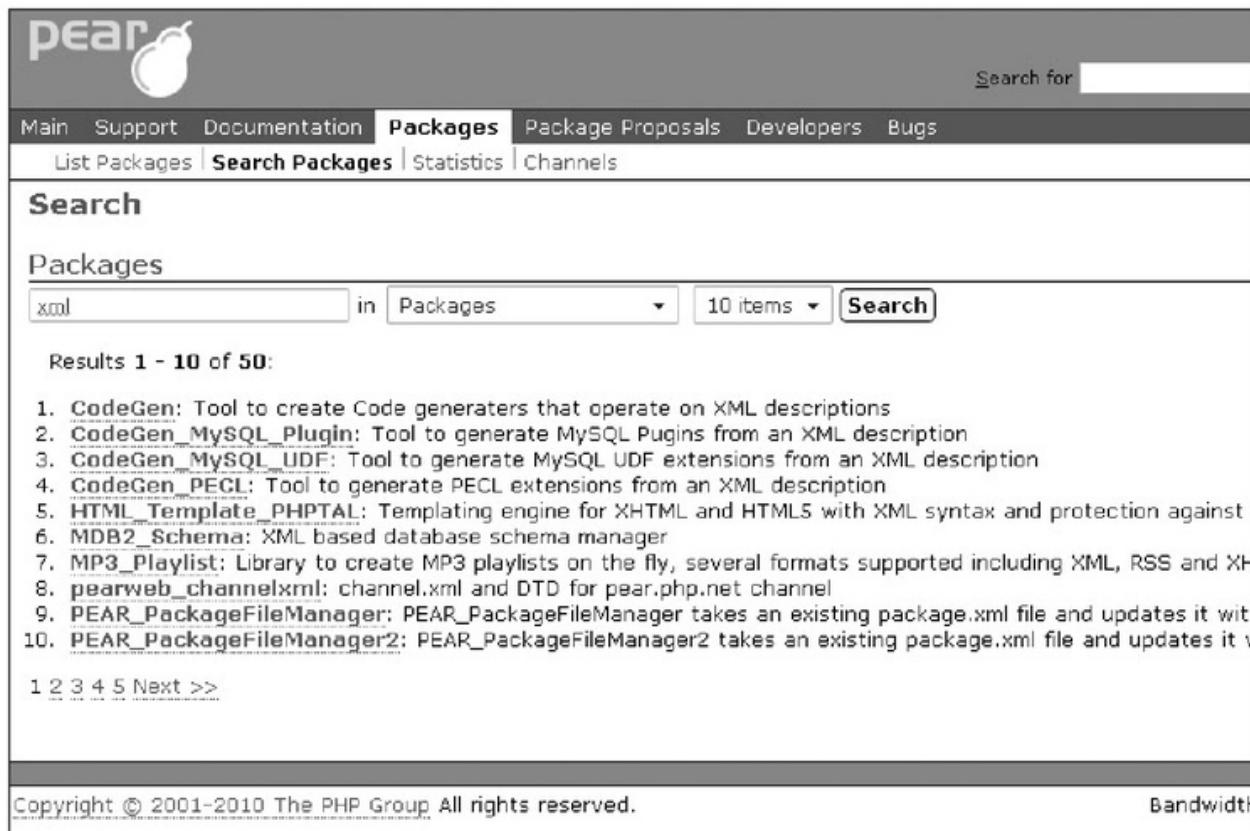
```

$xmlu->setOption(XML_UNSERIALIZER_OPTION_ATTRIBUTES_PARSE, true);
$xmlu->setOption(XML_UNSERIALIZER_OPTION_COMPLEXTYPE, 'array');
$xmlu->setOption(XML_UNSERIALIZER_OPTION_ENCODING_SOURCE, 'UTF-8');
$xmlu->setOption(XML_UNSERIALIZER_OPTION_ENCODING_TARGET, 'ISO-8859-1');

```

III SUCURSALES VIRTUALES

La empresa **Amazon** es una de las tantas que brindan acceso a su sitio a través de un servicio web, al permitir a los webmasters incluir listados de productos en sus sitios y recibir comisiones si la venta se concreta. En PEAR, contamos con el paquete **Services_Amazon**, que puede utilizarse previo registro en el sitio de la empresa.



The screenshot shows the PEAR website's search interface. The top navigation bar includes links for Main, Support, Documentation, Packages (which is highlighted in blue), Package Proposals, Developers, and Bugs. Below the navigation is a search bar labeled "Search for". Underneath the search bar are links for List Packages, Search Packages (also highlighted in blue), Statistics, and Channels.

The main content area has a title "Search" and a sub-section titled "Packages". A search form contains the query "xml" in the search field, the category "in Packages", a dropdown for "10 items", and a "Search" button. Below the search form, a message says "Results 1 - 10 of 50:" followed by a numbered list of 10 PEAR packages related to XML. At the bottom of the search results, there are page navigation links from 1 to 5 and a "Next >>" link.

At the very bottom of the page, there is a copyright notice "Copyright © 2001-2010 The PHP Group All rights reserved." and a "Bandwidth" link.

Figura 19. Los paquetes PEAR para manipular documentos XML son de gran utilidad para trabajar sobre respuestas obtenidas desde un servicio web.

Una vez hecho esto, convertimos el XML que acabamos de recuperar (**unserialize** y **getUnserializedData**) y, si no ocurrieron errores, mostramos el array resultante, como podemos ver en el código que está a continuación, que utiliza esta función:

```
$resultado = $xmlu->unserialize($respuesta->getBody());

if (!PEAR::isError($resultado)) {
    $resultado = $xmlu->getUnserializedData();
    print_r($resultado);
}
```



BENEFICIO PARA TODOS

Es importante saber que uno de los fines principales de los servicios web es el de lograr integrar aplicaciones para aumentar el valor agregado en las prestaciones brindadas tanto a los usuarios como a las compañías. De esta forma, se encargan de permitir la expansión de su funcionalidad hacia múltiples puntos a través de una red o también de una intranet.

```

Array
(
    [stat] => ok
    [photos] => Array
        (
            [page] => 1
            [pages] => 296
            [perpage] => 10
            [total] => 2959
            [photo] => Array
                (
                    [0] => Array
                        (
                            [id] => 5097534736
                            [owner] => 322982008N03
                            [secret] => 84ce616532
                            [server] => 4125
                            [farm] => 5
                            [title] => Beatrix
                            [ispublic] => 1
                            [isfriend] => 0
                            [isfamily] => 0
                        )
                    [1] => Array
                        (
                            [id] => 5093469155
                            [owner] => 525346208N04
                            [secret] => beff7aeb93
                            [server] => 4147
                            [farm] => 5
                            [title] => Fierce eyes
                            [ispublic] => 1
                            [isfriend] => 0
                            [isfamily] => 0
                        )
                    [2] => Array
                        (
                            [id] => 5093469143
                            [owner] => 525346208N04
                        )
                )
            )
        )
)

```

Figura 20. Las respuestas obtenidas desde Flickr pueden ser tratadas, previa conversión, como estructuras PHP tradicionales.

Lo que haremos con la información obtenida dependerá de la especificación de cada servicio y la utilidad que vayamos a darle pero, en el caso del ejemplo, Flickr permite mostrar imágenes mediante la siguiente sintaxis:

```
http://farm{farm-id}.static.flickr.com/{server-id}/{id}_{secret}.jpg
```

farm-id se obtiene desde:

\$result[photos][photo][indiceFoto][farm]

server-id se obtiene desde:

\$result[photos][photo][indiceFoto][server]



MUESTRAS GRATIS

Debemos recordar que podemos disponer de una gran cantidad de servicios web testeados para incluir en nuestras aplicaciones, estos se encuentran en los sitios www.soapware.org y www.xmethods.net, muchos de ellos son distribuidos en forma gratuita y pueden ser de utilidad para resolver necesidades frecuentes en cualquier desarrollo.

id se obtiene desde:

\$result[photos][photo][indiceFoto][id]

secret se obtiene desde: **\$result[photos][photo][indiceFoto][secret]**

Es interesante destacar que es posible incluir una imagen utilizando el código que se muestra a continuación, junto a un ejemplo de lo que lograremos:

```

```



Figura 21. La inclusión de las respuestas obtenidas desde Flickr en páginas HTML es directa y simple.

El sitio web Yahoo se encarga de ofrecernos algunas opciones que podemos utilizar para acceder a sus servicios mediante diversos protocolos, como por ejemplo: SOAP, REST y XML-RPC. De esta manera podremos obtener respuestas en varios formatos, como: JSON, XML, PHP, entre otros.

Debemos saber que es posible obtener información acerca de los servicios web ofrecidos por Flickr en el sitio www.flickr.com/services.

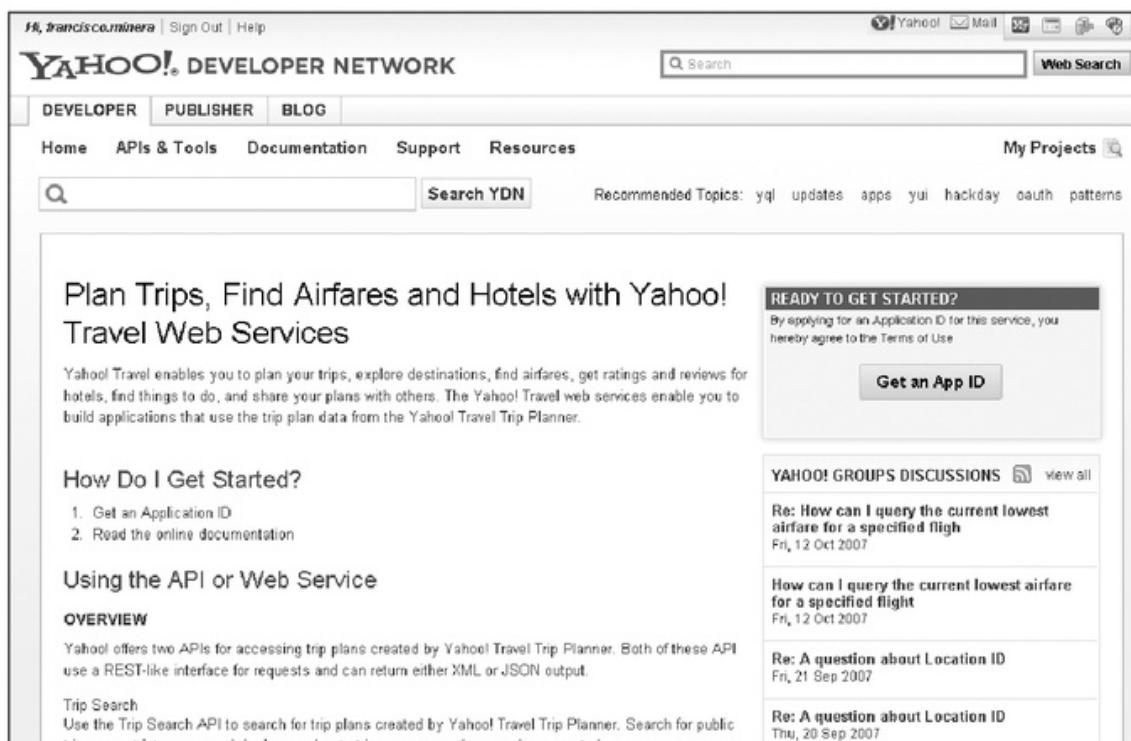


Figura 22. La empresa **Yahoo** dedica una sección especial de su sitio al desarrollo de servicios web que pueden ser accedidos desde distintos lenguajes de programación (<http://developer.yahoo.com>).

... RESUMEN

En este capítulo, realizamos un recorrido por los componentes fundamentales que constituyen un servicio web: capas intervinientes y protocolos base. Además, observamos las posibilidades que nos ofrece PHP tanto para acceder como para generar servicios, haciendo hincapié en la extensión soap y los paquetes PEAR disponibles.



ACTIVIDADES

PREGUNTAS TEÓRICAS

1 ¿Cuál es la utilidad básica de un servicio web?

2 ¿De qué manera beneficia a un desarrollo ya realizado?

3 ¿Qué es SOAP?

4 ¿Para qué sirve WSDL?

5 ¿Cuál es la función de UDDI?

6 ¿Cuáles son las capas componentes de un servicio web?

7 ¿Qué importancia da la industria a los servicios web?

8 ¿Cuál cree que será el futuro de los servicios web?

9 ¿En qué casos no utilizaría servicios web?
¿Por qué?

10 ¿Piensa que el desarrollo de servicios web es más sencillo que el de las aplicaciones tradicionales? ¿Por qué?

EJERCICIOS PRÁCTICOS

1 Acceda mediante PEAR al sitio www.flickr.com.

2 Genere un servicio web mediante la extensión soap.

3 Genere una aplicación que utilice Services_YouTube.

4 Genere una aplicación que utilice Services_W3C_HTML_Validator.

5 Genere una aplicación que utilice Services_Weather.

Orientación a objetos

A continuación, haremos un repaso por los principios del paradigma orientado a objetos, incluyendo las mejoras introducidas en las últimas versiones del lenguaje.

Antecedentes	198
Clases	199
Tipos de acceso a variables y funciones	200
Constantes	203
Constructores y destructores de clases	203
Subclases y forma de acceso	206
Captura de errores en clases, métodos y propiedades	207
Implementación de la herencia	209
Clases abstractas	213
Propiedades y métodos estáticos	215
Referenciación de objetos	218
Objetos como tipo de dato	221
Resumen	223
Actividades	224

ANTECEDENTES

La implementación de la orientación a objetos en PHP no tuvo un buen comienzo. Las características básicas no estaban desarrolladas, y esto limitaba a los programadores que conocían el paradigma, pero no contaban con la ayuda del lenguaje para generar aplicaciones sólidas. A partir de la versión 5, esto cambio, y el nuevo motor **Zend 2.0** permitió desarrollar aplicaciones veloces y consistentes.

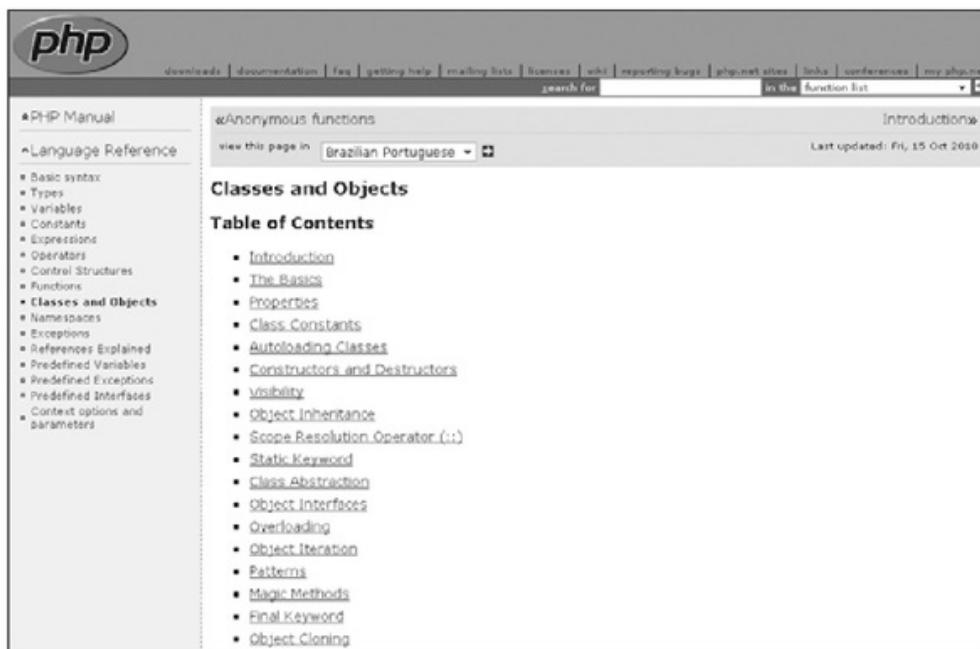


Figura 1. Las herramientas puestas a disposición del usuario en las últimas versiones del lenguaje sufrieron fuertes modificaciones.

Tomando como ejemplo lenguajes cuya implementación de la orientación a objetos es probadamente competente (como **Java**, **C#**, y **Python**), el equipo de desarrollo de PHP se propuso incorporar en las últimas versiones aquellas funcionalidades no incluidas y reelaborar las ya existentes para que los programadores generaran soluciones profesionales manteniendo siempre las características tradicionales del lenguaje, claro. Podemos obtener más información al respecto en el sitio oficial de PHP (www.php.net) donde se enumeran las características de la POO en las versiones más actuales.



MANTENER LA COMPATIBILIDAD

Como desarrolladores debemos tener en cuenta que conocer los cambios introducidos en el uso de la orientación a objetos resulta muy útil para mantener la compatibilidad de nuestras aplicaciones pensando en las futuras actualizaciones de los servicios de alojamiento web, o también para mejorar el desempeño de los desarrollos que realicemos.

CLASES

Podríamos definir a una **clase** como una plantilla o molde desde donde es posible generar objetos (instancias). La estructura de una clase contiene dos elementos básicos: variables y funciones (declaradas mediante **function**). La definición de una clase se delimita al iniciar una línea con la palabra **class**:

```
class nombreClase {
    public $nombreVariable;

    function nombreFuncion() {
        return $this->nombreVariable;
    }
}
```

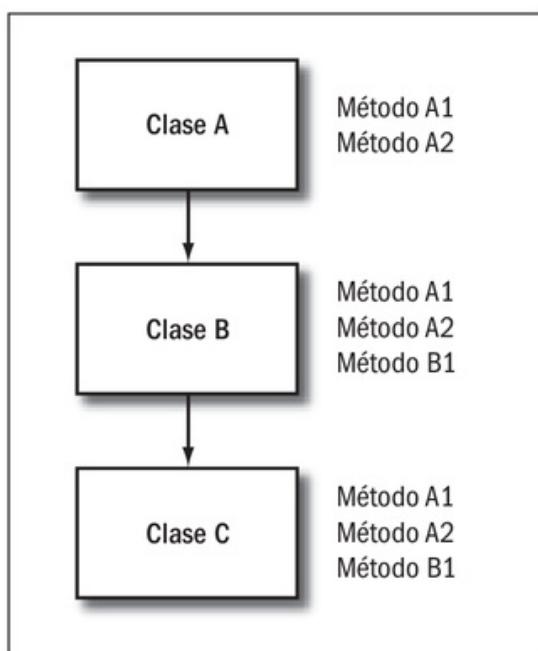


Figura 2. Las clases constituyen la pieza clave del paradigma orientado a objetos y pueden pensarse como plantillas para personalizar estructuras de datos.



EVOLUCIÓN EN LA POO

Devemos saber que en las últimas versiones de lenguaje de programación PHP, los cambios son notorios y pueden llegar a modificar la forma de desarrollo para los usuarios. La administración y el tratamiento de objetos es un ejemplo claro de esto, por eso se convierte en una de las características más reclamadas por los usuarios.

Cada instancia de una misma clase puede tomar valores particulares para sus variables aunque esto, como veremos, dependerá de los permisos sobre ellas.



Figura 3. La implementación del motor Zend 2.0 dio a las aplicaciones PHP mayor velocidad de ejecución y una gran estabilidad.

\$this es una variable reservada especial que contiene el objeto actual y se utiliza dentro del ámbito de una función definida dentro de una clase. Podemos utilizarla para referenciar a otras funciones o variables dentro de la misma clase. Otra opción equivalente a la palabra reservada **\$this** es utilizar **self**.

Tipos de acceso a variables y funciones

Tanto las variables (que dentro del ámbito de una clase se denominan **propiedades**) como las funciones (que dentro del ámbito de una clase se denominan **métodos**) admiten los llamados **modificadores de acceso**, entre los que se cuentan: **public**, **private**, y **protected**.

MODIFICADOR	DESCRIPCIÓN
public	La propiedad o método es accesible desde cualquier ámbito, incluyendo otras clases.
private	La propiedad o método sólo es accesible desde dentro de la clase a la que pertenece.
protected	La propiedad o método sólo es accesible desde dentro de la clase a la que pertenece o desde cualquier clase que derive de ella.

Tabla 1. Modificadores de acceso disponibles.

```

<?php
class nombreClase {

    public $variableA;
    protected $variableB;
    private $variableC;

    public function obtenerValor($v) {
        if ($v == 'A')
            return $this->variableA;
        else
            return 'Valor no disponible.';
    }
}

$c = new nombreClase();
echo $c->obtenerValor('A');

?>

```

```

119     {
120         if ($this->connection) {
121             $native_code = @mysql_errno($this->connection);
122             $native_msg = @mysql_error($this->connection);
123         } else {
124             $native_code = @mysql_errno();
125             $native_msg = @mysql_error();
126         }
127         if (is_null($error)) {
128             static $ecode_map;
129             if (empty($ecode_map)) {
130                 $ecode_map = array(
131                     1004 => MDB2_ERROR_CANNOT_CREATE,
132                     1005 => MDB2_ERROR_CANNOT_CREATE,
133                     1006 => MDB2_ERROR_CANNOT_CREATE,
134                     1007 => MDB2_ERROR_ALREADY_EXISTS,
135                     1008 => MDB2_ERROR_CANNOT_DROP,
136                     1022 => MDB2_ERROR_ALREADY_EXISTS,
137                     1044 => MDB2_ERROR_ACCESS_VIOLATION,
138                     1046 => MDB2_ERROR_NODSELECTED,
139                     1048 => MDB2_ERROR_CONSTRAINT,
140                     1049 => MDB2_ERROR_NOSUCHDB,
141                     1050 => MDB2_ERROR_ALREADY_EXISTS,
142                     1051 => MDB2_ERROR_NOSUCHTABLE,
143                     1054 => MDB2_ERROR_NOSUCHFIELD,
144                     1061 => MDB2_ERROR_ALREADY_EXISTS,
145                     1062 => MDB2_ERROR_ALREADY_EXISTS,
146                     1064 => MDB2_ERROR_SYNTAX,
147                     1091 => MDB2_ERROR_NOT_FOUND,
148                     1100 => MDB2_ERROR_NOT_LOCKED,
149                     1136 => MDB2_ERROR_VALUE_COUNT_ON_ROW,
150                     1142 => MDB2_ERROR_ACCESS_VIOLATION,
151                     1146 => MDB2_ERROR_NOSUCHTABLE,
152                     1216 => MDB2_ERROR_CONSTRAINT,
153                     1217 => MDB2_ERROR_CONSTRAINT,
154                     1356 => MDB2_ERROR_DIVZERO,
155                 );
156             }
157         }
158     }
159 }

```

The screenshot shows the EditPlus interface with the file 'index.php' open. The code block above is part of a larger class definition for a MySQL driver. It includes logic to map MySQL error codes to specific MDB2_ERROR constants, handling both connection-based and general cases. The code is annotated with line numbers from 119 to 135.

Figura 4. Los métodos de acceso nos permiten mantener un control detallado acerca de la utilización de las clases por parte del usuario.

El operador **new** nos permite crear una instancia de una clase.

Antes de la versión 5, todas las variables y las funciones eran públicas y se declaraban utilizando la palabra reservada **var**. Por cuestiones de compatibilidad hacia las versiones anteriores de PHP, esta opción sigue disponible para las propiedades y se considera un sinónimo para la palabra reservada **public**.

```
var $nombrePropiedad;
```

Los métodos, por su parte, no contaban con ningún modificador de acceso disponible, sólo se definían mediante un nombre y un conjunto de argumentos, esto como veremos a continuación también fue mejorado.

Es posible acceder tanto a partir de una instancia de una clase como desde dentro de la definición de la clase misma a las propiedades disponibles:

```
<?php

class nombreClase {
    public $nombrePropiedad;

    public function nombreMetodo() {
        echo "<li>".$this->nombrePropiedad;
    }
}

$c = new nombreClase();
echo "<li>".$c->nombrePropiedad;

?>
```

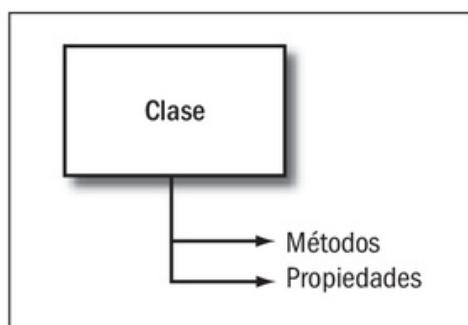


Figura 5. La orientación a objetos facilita el trabajo sobre todo en proyectos complejos que requieran actualizaciones.

Constantes

También es posible definir constantes dentro de una clase, mediante la palabra reservada **const**. A diferencia de las propiedades, no se les antepone el carácter \$ y siempre son de acceso público, sin poder determinar su acceso con modificadores:

```
<?php

class nombreClase1 {
    const nombreConstante = 'xxx';

    //...
}

echo "<li>".nombreClase1::nombreConstante;

?>
```

Veremos más acerca del operador :: en las próximas páginas.

Constructores y destructores de clases

Cada vez que se crea una instancia de una clase, se ejecuta una función llamada constructor. El nombre de esta función es **_construct** y podemos redefinirla si deseamos modificar su comportamiento por defecto:

```
<?php

class nombreClase {

    public $nombrePropiedad;
```



ENCAPSULACIÓN

Los modificadores de acceso pueden sernos útiles para implementar la encapsulación, un punto clave en la programación orientada a objetos más conocida como POO. Debemos saber que en las últimas versiones del lenguaje, se avanzó mucho sobre este tema, proporcionando a los desarrolladores varias opciones en reemplazo de las que ya existían.

```

function __construct() {
    $this->nombrePropiedad = 10;
}
}

$c = new nombreClase();
echo $c->nombrePropiedad;

?>

```

Antes de la versión 5, el nombre de la función constructora debía coincidir con el de la clase. Esto sigue teniendo validez en caso de que no definamos explícitamente la función **__construct**.

Así como existen los constructores, también están disponibles los destructores, que son métodos que se encargan de realizar las tareas que se necesitan ejecutar cuando un objeto ya no es referenciado por ninguna variable (por ejemplo liberar la memoria de forma definitiva). El nombre de esta función es **__destruct**, y podemos redefinirlo agregándole comportamientos personalizados:

```

<?php

class nombreClase {

    public $nombrePropiedad;

    function __construct() {
        $this->nombrePropiedad = 10;
    }

    function __destruct(){
        echo "<li>Destruyendo Objeto";
    }
}

$c = new nombreClase();

echo "<li>". $c->nombrePropiedad;

```

```
echo "<li>Termina el script";  
?  
?"
```

Salida:

```
<li>10  
<li>Termina el script  
<li>Destruyendo Objeto
```

El método destructor se invoca al final de un script. Puede ejecutarse explícitamente si eliminamos la instancia antes (el código de un destructor se interpreta cuando la instancia se elimina de la memoria):

```
<?php  
  
class nombreClase {  
  
    public $nombrePropiedad;  
  
    function __construct() {  
        $this->nombrePropiedad = 10;  
    }  
  
    function __destruct(){  
        echo "<li>Destruyendo Objeto";  
    }  
}  
  
$c = new nombreClase();  
  
echo "<li>".$c->nombrePropiedad;  
  
unset($c);  
  
echo "<li>Termina el script";  
?  
?"
```

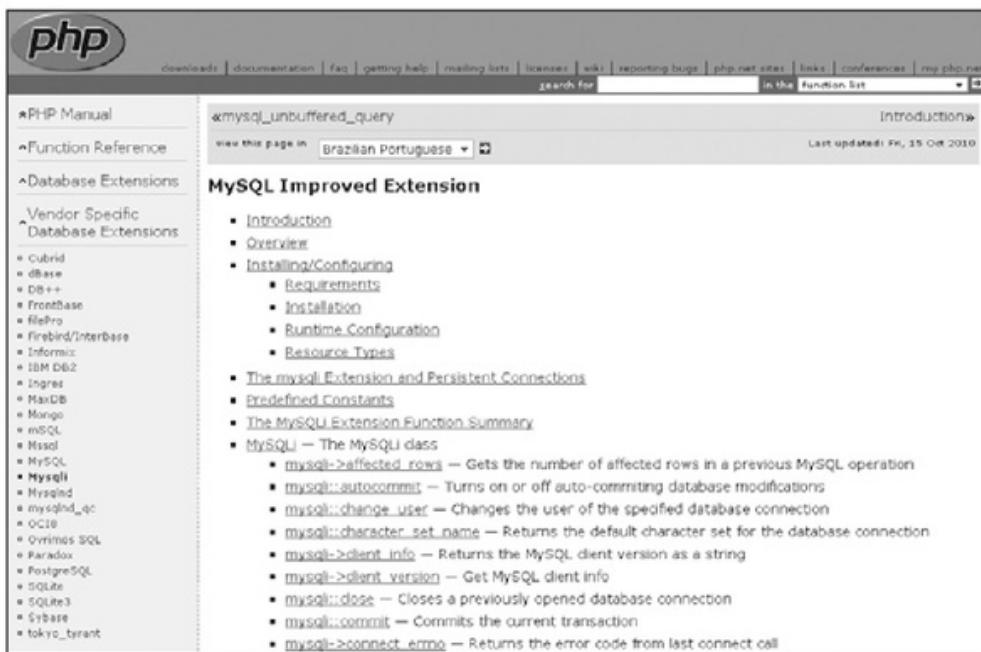


Figura 6. Cada vez son más las extensiones PHP que brindan su funcionalidad a través de clases, métodos y atributos.

Salida:

```
<li>10
<li>Destruyendo Objeto
<li>Termina el script
```

Subclases y forma de acceso

Una clase puede ser instanciada dentro de otra, por ejemplo:

```
<?php

class nombreClase1 {
    public $nombrePropiedad;

    function __construct() {
        $this->nombrePropiedad = new nombreClase2();
    }
}

class nombreClase2 {
    public function nombreMetodo() {
```

```

    return 'Valor retornado desde nombreClase2.';

}

}

$obj = new nombreClase1();

echo $obj->nombrePropiedad->nombreMetodo();

?>

```

Salida:

Valor retornado desde nombreClase2.

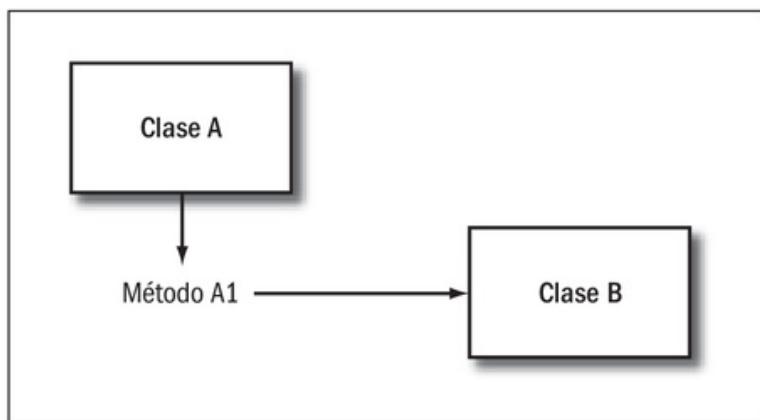


Figura 7. El acceso a una clase dentro de otra es una técnica frecuente para ligar las distintas partes de un sistema.

Realizar esta clase de tareas desde las versiones anteriores requería la utilización de variables temporales para almacenar los valores.

Captura de errores en clases, métodos y propiedades

Si intentamos invocar un método inexistente, por defecto recibiremos un mensaje de error. Para capturarlos, existe el método especial `__call` que se ejecuta de manera automática en esos casos. Recibe como argumentos automáticos el nombre del método y los parámetros pasados, en forma de array:

```
<?php
```

```
class nombreClase {  
  
    private $nombrePropiedad;  
  
    function __call($metodo, $atributos) {  
        return "Error: el metodo '$metodo' no esta definido.";  
    }  
}  
  
$c = new nombreClase();  
  
echo "<li>".$c->nombreMetodo();  
  
?>
```

En el mismo sentido, los métodos **__set** y **__get** serán invocados en caso de intentar asignar o recuperar respectivamente valores de propiedades no declaradas:

```
<?php  
  
class nombreClase {  
    private $nombrePropiedad;  
  
    private function __get($propiedad) {  
        return "Error: no es posible leer '$propiedad.'";  
    }  
  
    private function __set($propiedad, $valor) {  
        echo "<li>Error: no es posible modificar '$propiedad.'";  
    }  
}  
  
$c = new nombreClase();  
  
$c->variableNoExistente = 101;  
  
echo "<li>".$c->variableNoExistente;  
  
?>
```



Figura 8. El control de acceso a propiedades nos permite asegurar el correcto funcionamiento de una clase y personalizar los mensajes de error.

Así como disponemos de `__call` para métodos, y `__set` y `__get` para propiedades, la función especial `__autoload`, en caso de ser definida de manera explícita, sería invocada de forma automática si intentáramos utilizar clases que no han sido definidas:

```
<?php

function __autoload($nombreClase) {
    echo "<li>La clase '$nombreClase' no esta definida.";
    exit;
}

$c = new claseInexistente();

?>
```

Implementación de la herencia

Un concepto importante en la orientación a objetos es el de herencia: cuando una clase deriva de otra automáticamente hereda sus propiedades y métodos. En su propia definición, puede modificar valores y redefinir métodos, aunque esto puede evitarse, como veremos. Para que una clase descienda de otra, se utiliza la palabra reservada `extends`:

```
<?php

class nombreClaseA {
    public $nombrePropiedad;
```

```

function __construct() {
    $this->nombrePropiedad = 10;
}

public function nombreMetodo() {
    return 'A';
}
}

class nombreClaseB extends nombreClaseA {
    public $nombrePropiedad;

    public function nombreMetodo() {
        return 'B';
    }
}

$A = new nombreClaseA();
echo "<li>".$A->nombreMetodo();
echo "<li>".$A->nombrePropiedad;

$B = new nombreClaseB();
echo "<li>".$B->nombreMetodo();
echo "<li>".$B->nombrePropiedad;

?>

```

Salida:

```
<li>A
```



AUTOCARGA DE CLASES

La función especial denominada **__autoload** puede sernos de mucha utilidad para implementar lo que se llama **autocarga de clases** que, en resumidas cuentas, se encarga de permitirnos dinamizar la carga de clases definidas en archivos externos y de esta forma ubicar dentro de la definición de la función una estructura de control.

```
<li>10  
<li>B  
<li>10
```

Como vimos, según el modificador de acceso asignado a un método, las clases derivadas podrán o no redefinir su funcionalidad. El modificador **final** anula esta posibilidad por completo, declarando una función que no puede ser reescrita:

```
<?php  
  
class nombreClaseA {  
    public $nombrePropiedad;  
  
    function __construct() {  
        $this->nombrePropiedad = 10;  
    }  
  
    final function nombreMetodo() {  
        return 'A';  
    }  
}  
  
class nombreClaseB extends nombreClaseA {  
    public $nombrePropiedad;  
  
    public function nombreMetodo() {  
        return 'B';  
    }  
}  
  
$A = new nombreClaseA();  
echo "<li>".$A->nombreMetodo();  
echo "<li>".$A->nombrePropiedad;  
  
$B = new nombreClaseB();  
echo "<li>".$B->nombreMetodo();  
echo "<li>".$B->nombrePropiedad;  
  
?>
```

Esto produce como resultado un error fatal:

```
Cannot override final method nombreClaseA::nombreMetodo()
```

Relacionado con la extensión de clases ya declaradas, podemos citar a las llamadas **interfaces**, que permiten definir el conjunto de métodos que deberán determinar (de manera obligatoria) las clases que la implementan. Esta opción se introdujo en PHP versión 5, y podemos ponerla en práctica utilizando la palabra reservada **interface**:

```
<?php

interface nombreInterfase {
    public function nombreMetodo1($nombreArgumento);
    public function nombreMetodo2();
}

class nombreClase1 implements nombreInterfase {
    function nombreMetodo1($nombreArgumento) {
        return "<li>Implementacion de nombreMetodo1 en nombreClase1";
    }

    function nombreMetodo2() {
        return "<li>Implementacion de nombreMetodo2 en nombreClase1";
    }
}

class nombreClase2 implements nombreInterfase {
    function nombreMetodo1($nombreArgumento) {
        return "<li>Implementacion de nombreMetodo1 en nombreClase2";
    }

    function nombreMetodo2() {
        return "<li>Implementacion de nombreMetodo2 en nombreClase2";
    }
}

$obj1 = new nombreClase1;
echo $obj1->nombreMetodo1(1);
echo $obj1->nombreMetodo2();
```

```
$obj2 = new nombreClase2;
echo $obj2->nombreMetodo1(2);
echo $obj2->nombreMetodo2();

?>
```

La salida es:

```
<li>Implementacion de nombreMetodo1 en nombreClase1
<li>Implementacion de nombreMetodo2 en nombreClase1
<li>Implementacion de nombreMetodo1 en nombreClase2
<li>Implementacion de nombreMetodo2 en nombreClase2
```

Dentro de la composición de una interfaz, no es posible definir métodos, sólo podemos declararlos (únicamente sus nombres y argumentos correspondientes). Todos los métodos en una interfaz deben tener un nivel de acceso público. Las clases pueden implementar más de una interfaz, separando cada nombre mediante comas.

Clases abstractas

Otro modificador de acceso es **abstract**, que permite definir clases y métodos como abstractos. Una clase se reconoce como abstracta cuando no puede ser instanciada de manera directa. La finalidad de este tipo de clases es la de servir como base para otras que sí podrán ser instanciadas (utilizando **extends**):

```
<?php

abstract class nombreClase1 {
    //...
```



HERENCIA MÚLTIPLE

Debemos saber que en muchos lenguajes, está disponible la llamada **herencia múltiple**, la cual consiste en la posibilidad de que una clase derive de otras (más de una). Esta opción no está disponible en el lenguaje de programación PHP, pero podemos suplirla utilizando interfaces o también realizando derivaciones escalonadas.

```
}

class nombreClase2 extends nombreClase1 {
    //...
}

$obj = new nombreClase2();

?>
```

El caso de los métodos abstractos es similar: no se pueden invocar directamente desde una instancia de la clase abstracta, pero pueden ser heredados por las clases que la extienden y ser utilizados desde allí.

```
<?php

abstract class nombreClase1 {
    //...

    abstract function nombreMetodoAbstracto() {
        //...
    }
}

class nombreClase2 extends nombreClase1 {
    //...
}

$obj = new nombreClase2();
```



DEFINICIÓN DE MÉTODOS

En nuestro trabajo como desarrolladores, es posible que en algún caso particular necesitemos en ciertas situaciones definir algunos métodos en las clases derivadas y no en la clase abstracta. Para resolver este eventual requerimiento, podemos recurrir a realizar la declaración de métodos abstractos o a las interfaces.

```
?>
```

Una clase derivada de una clase abstracta puede ser a su vez abstracta.

Propiedades y métodos estáticos

Además de los modificadores de acceso, otra posibilidad de cambiar el comportamiento de las propiedades y de los métodos es a través de la opción **static**, que nos permite mantener los valores sin depender de una instancia en particular:

```
<?php

class nombreClase {

    private static $nombrePropiedad = 101;

    public static function nombreMetodo() {
        return nombreClase::$nombrePropiedad;
    }
}

echo "<li>".nombreClase::nombreMetodo();

?>
```

Existe una sintaxis alternativa a través del operador **::**, que puede ser utilizado tanto desde fuera de la clase como desde dentro de un método:

```
<?php

class nombreClase {
    static $nombrePropiedad = 101;

    public function nombreMetodo() {
        return nombreClase::$nombrePropiedad;
    }
}
```

```
echo "<li>".nombreClase::$nombrePropiedad;  
  
$c = new nombreClase();  
  
echo "<li>".$c->nombreMetodo();  
  
?>
```

Incluso también podemos invocar métodos:

```
<?php  
  
class nombreClase {  
  
    static $nombrePropiedad = 101;  
  
    static function nombreMetodo() {  
        return nombreClase::$nombrePropiedad;  
    }  
}  
  
echo "<li>".nombreClase::nombreMetodo();  
  
?>
```

No es posible acceder desde un objeto a los atributos estáticos de la clase:

```
<?php
```



ARGUMENTOS POR VALOR Y POR REFERENCIA

En las versiones del lenguaje de programación PHP, anteriores a la 5, la tarea de pasar algunos objetos como argumento a funciones o métodos se realizaba por valor, a menos que se indicara explícitamente lo contrario utilizando el operador & (ampersand). Pero es importante saber que en PHP 5 y superiores son pasados por referencia.

```

class nombreClase {

    static $nombrePropiedad = 101;

    static function nombreMetodo() {
        return nombreClase::$nombrePropiedad;
    }
}

$c = new nombreClase();

echo "<li>".$c->nombrePropiedad;

?>

```

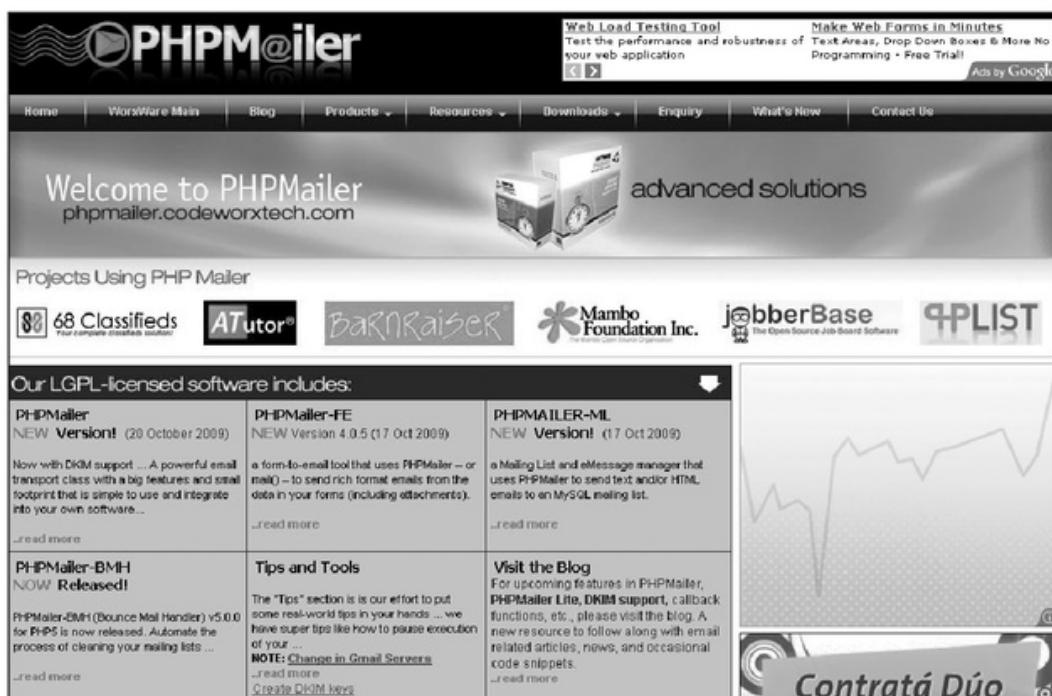


Figura 9. Cada vez son más las aplicaciones PHP que han sido desarrolladas y diseñadas utilizando clases.

Con esta sintaxis, tampoco es posible acceder a una propiedad no estática desde un método estático:

```
<?php
```

```

class nombreClase {

    public $nombrePropiedad = 101;

    static function nombreMetodo() {
        return nombreClase::$nombrePropiedad;
    }
}

echo "<li>".nombreClase::nombreMetodo();

/*
Access to undeclared static property: nombreClase::$nombrePropiedad
*/
?>

```

Referenciación de objetos

Otra de las características que difieren en la implementación de la POO en las últimas versiones del lenguaje es la relacionada con la referenciación de objetos. En PHP versión 5 y superiores, los objetos son referenciados a través de manejadores, por lo que una operación como la siguiente asignaría a **\$obj2** una referencia al mismo objeto:

```

<?php

$obj1 = new nombreClase();
$obj2 = $obj1;

?>

```

Esto deriva en que cualquier operación sobre **\$obj1** afectará las características de **\$obj2**, y viceversa. Por lo tanto, lo que se crea no es otro objeto, sino un alias o referencia en memoria de él mismo. Si lo que necesitamos es clonar un objeto para que éste pueda luego mantener un comportamiento propio independiente de los demás, podemos utilizar la función **clone**, que de forma predeterminada crea un objeto copiando todos los atributos del original:

```
<?php

class nombreClase {

    public $nombrePropiedad;

    public function nombreMetodo() {
        return $this->nombrePropiedad;
    }

}

$obj1 = new nombreClase();

$obj1->nombrePropiedad = 101;

$obj2 = clone $obj1;

$obj2->nombrePropiedad = 102;

echo "<li>".$obj1->nombrePropiedad;

echo "<li>".$obj2->nombrePropiedad;

?>
```

La salida es:

```
<li>101
<li>102
```



MODIFICACIÓN DE ARGUMENTOS

Es importante saber que el paso de objetos por referencia permite que las modificaciones realizadas tengan una repercusión directa sobre el objeto original. El comportamiento por defecto en lo referido al paso de variables por valor o por referencia cambió en las últimas versiones del lenguaje, por esta razón debemos estar atentos.

En PHP 4, la simple asignación implicaba una clonación:

```
$obj2 = $obj1;
```

Y para duplicar el objeto completo se utilizaba el operador & para definir la clonación de forma explícita, de la siguiente manera:

```
$obj1 =& new nombreClase();
```

```
$obj2 =& $obj1;
```

Volviendo a PHP 5 y superiores, es posible redefinir la función reservada `__clone` para personalizar la clonación agregando los comportamientos deseados, como en el ejemplo que vemos a continuación:

```
<?php

class nombreClase {

    public $nombrePropiedad;

    public function nombreMetodo() {
        return $this->nombrePropiedad;
    }

    function __clone() {
        //... código ...
    }
}

?>
```

El código dentro de la definición de `__clone` se ejecuta inmediatamente después de realizar la clonación, copiando los valores del objeto al nuevo.

Para comparar objetos, tenemos dos opciones: utilizar el operador tradicional de comparación simple (==) o el operador de identidad (===). En el primer caso,

el resultado será verdadero si ambos objetos tienen los mismos atributos y cada uno de ellos posee los mismos valores (comparación por valor).

Si por el contrario utilizamos el operador idéntico, el resultado será verdadero si las variables comparadas son referencias a la misma instancia de la misma clase, es decir si apuntan a la misma dirección de memoria.

Objetos como tipo de dato

En las versiones anteriores, los objetos eran tratados como estructuras simples (arrays o cadenas de caracteres) mientras que ahora es posible utilizar determinadas clases para referenciar un tipo de dato específico, por ejemplo:

```
<?php

class nombreClase1 {
    //...
}

class nombreClase2 {
    //...
}

function x(nombreClase1 $objeto) {
    //...
}

$obj1 = new nombreClase1();

$obj2 = new nombreClase2();

x($obj1);
```

III REFLEXIÓN

A partir de la versión 5 del lenguaje de programación PHP, se incluye un API que permite la ingeniería inversa de clases, interfaces y métodos [además permite recuperar los comentarios adjuntos]. Podemos encontrar más información acerca de este tema en el sitio oficial de PHP, que se encuentra en la dirección web <http://ar.php.net/reflection>.

```
x($obj2);
?>
```

La segunda llamada a **x** produce un error, ya que la función esperaba un objeto de tipo **nombreClase1**:

```
x($obj2);
```

En caso de que una función o método acepte un objeto de un determinado tipo, también recibirá instancias de las clases derivadas.

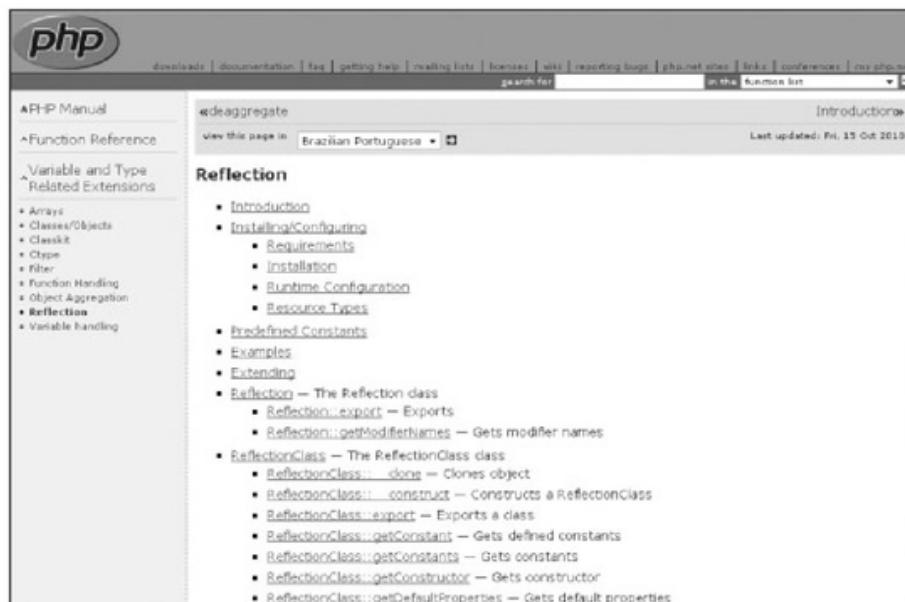


Figura 10. La reflexión es una de las técnicas incluidas dentro de las últimas versiones del lenguaje.

En relación con este tema, contamos con el operador **instanceof**, que permite saber si un objeto es una instancia de una determinada clase:

```
<?php

class nombreClase1 {
    //...
}

class nombreClase2 {
    //...
```

```
}

$obj1 = new nombreClase1();

$obj2 = new nombreClase2();

if ($obj1 instanceof nombreClase1) {
    echo "<li> Este objeto pertenece a nombreClase1.";
} else {
    echo "<li> Este objeto no pertenece a nombreClase1.';

}

if ($obj2 instanceof nombreClase1) {
    echo "<li> Este objeto pertenece a nombreClase1.";
} else {
    echo "<li> Este objeto no pertenece a nombreClase1.';

}

?>
```



RESUMEN

Un avance esperado por muchos es, sin duda, el referido a la nueva forma de implementar la programación orientada a objetos. En este capítulo, dimos un recorrido por las características salientes de esta técnica de programación y mostramos en detalle las nuevas posibilidades incluidas. Además, vimos a través de ejemplos las diferencias respecto de las versiones anteriores del lenguaje en relación con este tema.



ACTIVIDADES

PREGUNTAS TEÓRICAS

1 ¿Cuál cree que es la principal ventaja de la POO?

2 ¿Qué cambio valora más en relación con las versiones anteriores a la 4?

3 ¿Qué cree que falta?

4 ¿Piensa que utilizar este paradigma facilita la programación?

5 ¿Por qué cree que la POO es tan popular?

6 ¿Alguna vez utilizó en sus desarrollos esta técnica?

7 Dé ejemplos de sistemas POO.

8 ¿Qué es un modificador de acceso?

9 ¿Para qué sirve la herencia?

10 ¿Qué es una propiedad estática?

EJERCICIOS PRÁCTICOS

1 Genere un libro de visitas mediante POO.

2 Desarrolle una clase para realizar consultas a una base de datos MySQL.

3 Implemente un carrito de compras para almacenar los pedidos de los usuarios.

4 Programe un mismo sistema utilizando POO y programación a través del método procedural.

5 Piense un ejemplo de uso de herencia entre clases.

Seguridad en aplicaciones web

Las características generales de una aplicación a la que se accede a través de Internet demandan necesariamente la exigencia de planear un esquema de seguridad para evitar ataques y preservar su integridad. En este capítulo, presentaremos algunas de las herramientas provistas por PHP al respecto.

Introducción	226
Filtrado de datos de entrada y salida	227
Formularios	230
CAPTCHA	234
Almacenar información en base de datos	241
El caso de MySQL	251
Seguridad en bases de datos	252
Inclusión de archivos	255
Ajax	256
Línea de comandos	257
Register Globals	258
Modo seguro en PHP	260
Manipulación del archivo php.ini	262
Seguridad en PHP 6	263
Resumen	263
Actividades	264

INTRODUCCIÓN

Al desarrollar un sistema, debemos considerar que, eventualmente, accederán a él usuarios con diversas intenciones, por lo que la seguridad no tiene que ver con la cantidad de personas que accedan o con el tipo de servicio que se brinda.

PHP mantiene un sitio dedicado al tema en donde podremos encontrar recursos y artículos para comenzar a delinejar un plan de seguridad acorde con nuestras necesidades: <http://phpsec.org/library>.

En las próximas páginas, trataremos algunos de los temas más relevantes en cuanto a soluciones referidas a la seguridad en aplicaciones web utilizando PHP:

The screenshot shows the official PHP website at <http://php.net/>. The main navigation bar includes links for downloads, documentation, faq, getting help, mailing lists, licenses, viva, reporting bugs, php.net sites, links, conferences, and my.php.net. A search bar is present. On the right, there's a sidebar titled "Stable Releases" listing "Current PHP 5.3 Stable: 5.3.3" and "Current PHP 5.2 Stable: 5.2.13". Below it is a section for "Upcoming Events [add]" with a link to "October". Under "Conferences", there's a link to "29. PHP Barcelona Conference 2010". The main content area features a banner for "Upcoming conferences: Zend / PHP Conference, PHP Barcelona Conference 2010". Below this, a box for "PHP 5.3.3 Released!" contains the release date "[22 Jul 2010]", a brief description of the improvements, and a note about backwards incompatible changes. It includes a code snippet illustrating a namespace-related change. Further down, there's a section for "Security Enhancements and Fixes in PHP 5.3.3" with a bullet point about rewrote var_export(). The left sidebar has sections for "What Is PHP?", "Upcoming conferences", "Thanks To", and a list of sponsors.

Figura 1. Las últimas versiones de PHP incluyen avances en relación con las funcionalidades para implementar sitios seguros.

- Filtrado de datos.
- Formularios.
- Bases de datos.
- Encriptación.
- Inclusión de archivos.
- Aplicaciones Ajax.
- Funciones de la línea de comandos.
- *Register globals* y modo seguro.
- Cambios en PHP versión 6.

Estos temas son aplicables a toda clase de sistemas, sin importar su tamaño ni la complejidad de las tareas que éste realiza.

FILTRADO DE DATOS DE ENTRADA Y SALIDA

Uno de los conceptos más importantes es el referido a la validación de datos que ingresan en la aplicación desde el exterior. Casos como el envío de formularios, la carga de archivos o el acceso a servicios web son sólo algunos ejemplos usuales de entradas externas que deben ser tratadas y normalizadas para preservar la integridad de las aplicaciones, como veremos en este capítulo.

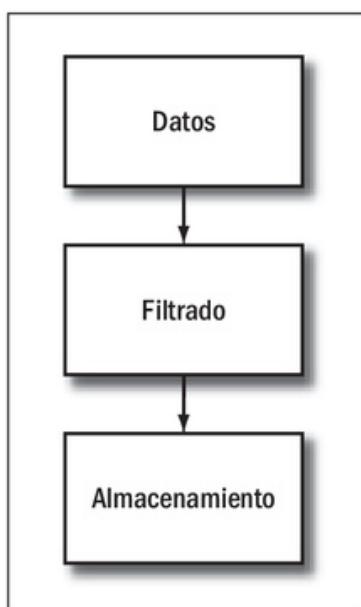


Figura 2. Independientemente de su alcance, toda aplicación debe validar el ingreso de datos externos antes de procesarlos.

Las validaciones se implementan sobre la base de reglas: si un dato cumple con determinadas pruebas, es válido, caso contrario, debe producirse un corte en la ejecución del script e informar lo sucedido al usuario.

PHP provee una serie de funciones para escapar datos, entre la que encontramos:

htmlentities: convierte caracteres especiales a las entidades HTML correspondientes. Recibe como argumentos la cadena por convertir, el tratamiento que se dará a las comillas simples y dobles (opcional, **ENT_COMPAT** para convertir las dobles y preservar las simples, **ENT_QUOTES** para convertir ambas y **ENT_NOQUOTES** para preservarlas, por defecto **ENT_COMPAT**), y el juego de caracteres (opcional, por ejemplo ISO-8859-1, ISO-8859-15, UTF-8, cp866, cp1251, cp1252, KOI8-R, BIG5, GB2312, BIG5-HKSCS, Shift_JIS, o EUC-JP, por defecto ISO-8859-1).

Si por ejemplo alguna de las variables contenía código JavaScript perjudicial, esto evitará que el navegador lo interprete como tal:

```
<?php

$cadena = '<SCRIPT LANGUAGE="JavaScript">';
$cadena .= '//codigo ...';
$cadena .= '</SCRIPT>';

echo htmlentities($cadena);

?>
```

La salida sería:

```
&lt;SCRIPT LANGUAGE="JavaScript"&gt;//codigo ...&lt;/SCRIPT&gt;
```

Podemos obtener la traducción completa a partir de la función `get_html_translation_table` (utilizando la constante `HTML_ENTITIES`), que devuelve un array con los caracteres origen y destino:

```
<?php

echo '<pre>';
print_r(get_html_translation_table(HTML_ENTITIES));
echo '</pre>';

?>
```

Por su parte, la función `html_entity_decode` produce el efecto inverso: convierte desde entidades hacia los caracteres correspondientes y recibe los mismos argumentos, es decir, posee la misma interfaz que el metodo anterior.



ACCESO POR URL

La activación de la directiva denominada `use_trans_sid` implica que el identificador de sesión (que podemos obtener mediante la función llamada `session_id`) será agregado a la URL actual. Debemos tener en cuenta que esto puede llegar a ser un punto vulnerable dentro de una aplicación, sobre todo en sistemas no desarrollados correctamente.

La función **htmlspecialchars** es idéntica a **htmlentities** sólo que traduce ciertos caracteres especiales de Internet: *ampersand* (& por &), comillas dobles (" por "), comillas simples (' por '), menor que (< por <) y mayor que (> por >). Recibe los mismos argumentos.



Figura 3. En general, no hará falta que descarguemos clases externas: la distribución base de PHP incorpora extensiones para los aspectos relacionados con la seguridad.

La traducción completa puede obtenerse a partir de la función **get_html_translation_table** (utilizando la constante **HTML_SPECIALCHARS**), que devuelve un array con los caracteres de origen y destino, de la siguiente manera:

```
<?php

echo '<pre>';
print_r(get_html_translation_table(HTML_SPECIALCHARS));
echo '</pre>';

?>
```

Más adelante veremos en este mismo capítulo cómo existen implementaciones similares específicamente orientadas al ingreso de registros en una base de datos o a su consulta para escapar las informaciones almacenadas. A su vez, mostraremos diversas maneras de evitar ataques provocados por el usuario al ingresar datos, de manera voluntaria e involuntaria, y evitando así agujeros de seguridad importantes.

FORMULARIOS

Es común que las aplicaciones web interactúen con datos externos, ingresados posiblemente por los mismos usuarios que las utilizan en forma de archivos o ficheros. Los formularios son un claro ejemplo de esta situación.

La información puede ingresar en una aplicación principalmente a través de dos métodos: **GET** (por URL) y **POST**. En el primero, el destino del formulario por lo general permitirá al usuario de la aplicación visualizar las variables enviadas en la ruta de la página, y también modificar sus valores para ver qué sucede. Si bien las variables enviadas a través de **POST** pueden ser observadas en el código fuente del formulario, la exposición es menor. El envío de archivos es otro punto que puede ser validado utilizando PHP. El atributo **enctype** del elemento **form** indica que el formulario es apto para enviar archivos:

```
<form method="POST" enctype="multipart/form-data">
```

El usuario podrá seleccionar uno a través del elemento **file**:

```
<input type="file" id="archivo" name="archivo" />
```

El tamaño máximo en bytes puede indicarse mediante el elemento **MAX_FILE_SIZE**:

```
<input type="hidden" name="MAX_FILE_SIZE" value="1024" />
```

A pesar de que el array **\$_FILES** contiene información acerca de los archivos enviados, no dice nada respecto del origen de ellos. Por el lado de PHP, existen diversas funciones para validar esto, como veremos a continuación.

- **is_uploaded_file** recibe el nombre de un archivo (que puede obtenerse desde el array **\$_FILES**) y devuelve verdadero si fue subido a través de un formulario.

```
<?php  
  
$archivo = $_FILES['archivo']['tmp_name'];  
  
if (is_uploaded_file($archivo)) {
```

```

$_SESSION['archivoSubido'] = true;
}

?>

```



Figura 4. Uno de los puntos más importantes y comunes en referencia al ingreso de datos es a través de formularios.

- **move_uploaded_file** recibe la ruta origen a un archivo y lo copia a una ruta destino, siempre y cuando haya sido enviado mediante un formulario. Aquí podemos ver un pequeño ejemplo de su funcionamiento:

```

<?php

$origen = $_FILES['archivo']['tmp_name'];
$destino = '/imagenes/' . $_FILES['archivo']['name'];

if (move_uploaded_file($origen, $destino)) {
    $_SESSION['archivoCopiado'] = true;
}

?>

```

III URLENCODE

La función **urlencode** permite codificar una cadena que se utilizará como parte de una URL. El valor returned es la cadena original con todos los caracteres no alfanuméricos (excepto -, _, y .) reemplazados con un signo de porcentaje (%) seguido por dos dígitos hexadecimales. Los espacios son codificados como signos de suma (+).

Para verificar el tamaño, está disponible la función **filesize**, que recibe como argumento la ruta al archivo y lo devuelve luego para su comparación:

```
<?php

$archivo = $_FILES['archivo']['tmp_name'];

if (filesize($archivo) <= 1024) {
    $_SESSION['tamanioCorrecto'] = true;
}

?>
```

Una de las formas de validar el origen de un formulario es usar variables de sesión:

```
<?php

session_start();

$_SESSION['identificador'] = md5(uniqid());

?>

<form method="POST">
<input type="hidden" id="identificador" name="identificador" value=" <?php
echo $_SESSION['identificador']; ?>" />

...
<input type="submit" value="enviar" />
</form>
```

En la página receptora se compara el valor de la variable enviada a través del formulario con el de la variable:

```
<?php

session_start()
```

```

if (isset($_SESSION['identificador']) &&
    $_POST['identificador'] == $_SESSION['identificador']) {
    //procesar formulario
}

?>

```

La función **uniqid** devuelve un identificador único basado en la hora actual en microsegundos, mientras que la función **md5** encripta la cadena recibida como argumento aplicándole el algoritmo de encriptación del mismo nombre (veremos más acerca de esto en este mismo capítulo).

Los formularios poseen un atributo **action** que permite establecer la dirección a la cual se enviarán. PHP dispone de un array especial llamado **\$_SERVER**, que incluye además al índice **HTTP_REFERER**. Su valor indica en teoría qué página solicitó la actual, sin embargo, puede ser reemplazado por otro (mediante sockets, por ejemplo), por lo que no es recomendable utilizarlo.

Es posible generar scripts para imitar el envío de datos a través de formularios que utilizan el método **POST** usando las cabeceras necesarias para que el navegador reconozca la petición. Veamos un ejemplo:

```

<?php

$contenido[] = "x=$x";
$contenido[] = "y=$y";
$contenido[] = "z=$z";

$contenido = implode("&", $contenido);

$logitud = strlen($contenido);

$peticion .= "POST /pagina.php HTTP/1.1\r\n";
$peticion .= "Host: servidor.com\r\n";
$peticion .= "Content-Type: application/x-www-form-urlencoded\r\n";
$peticion .= "Content-Length: $logitud\r\n";
$peticion .= "Connection: close\r\n";
$peticion .= "\r\n";
$peticion .= $contenido;

```

```

if ($manejador = fsockopen('servidor.com', 80)) {
    fputs($manejador, $peticion);

    while (!feof($manejador)) {
        $respuesta .= fgets($manejador, 1024);
    }

    fclose($manejador);
}

?>

```

Los datos obtenidos podrían utilizarse para comprobar la respuesta del servidor.

CAPTCHA

Una de las opciones para incluir es **CAPTCHA** (*Completely Automated Public Turing test to tell Computers and Humans Apart*, Prueba de Turing pública y automática para diferenciar máquinas y humanos), una técnica que permite agregar más seguridad al envío de formularios. Ha adquirido gran popularidad gracias a la existencia de extensiones para generar gráficos de forma dinámica, incluyendo en ellos caracteres alfanuméricos (**GD**, sobre el cual podemos encontrar más información en el sitio oficial de PHP en la sección: www.php.net/image).

The screenshot shows a Google Account password recovery form. At the top, there are fields for 'Re-enter password' and checkboxes for 'Stay signed in' and 'Enable Web History'. Below that is a 'Security Question' dropdown set to 'Choose a question ...' with a note about recovering the password if forgotten. There is also a 'Learn More' link. The 'Answer' field is empty. Underneath is a 'Recovery email:' field containing a placeholder email address. A note explains that this is used for account authentication if problems occur or the password is forgotten, with a 'Learn More' link. The 'Location:' dropdown is set to 'Argentina'. Below that is a 'Word Verification:' field with a CAPTCHA image showing the text 'dingistar' and a text input field for re-typing it. A note states that letters are not case-sensitive. At the bottom, there is a 'Terms of Service' checkbox with a note about checking account information and reviewing terms. Another note mentions that with Gmail, users won't see blinking banner ads. A 'Printable Version' link is at the very bottom right.

Figura 5. Cada vez son más los sitios que incorporan validaciones CAPTCHA en sus páginas.

Resulta muy común en aplicaciones como foros, encuestas y envío de correo. La idea básica es agregar, luego de los campos correspondientes al nombre, apellido, etcétera, una imagen que contenga caracteres y un cuadro de texto para que el usuario que completa el formulario ingrese lo visualizado. Si el texto no coincide con lo exhibido por la imagen, se emite un mensaje de error y se discontiúa el envío de datos.

Aunque hay maneras de saltar este paso (por ejemplo a partir de una aplicación **OCR**, o sea de reconocimiento óptico de caracteres, similar a los utilitarios de los escáneres), es un punto más a favor de la no automatización en el envío de formularios. Actualmente, no existe una manera sistemática para reconocer los caracteres de la imagen a través de una aplicación.

Entre las muchas implementaciones, está disponible una de PEAR (<http://pear.php.net>) llamada **Text_CAPTCHA**.

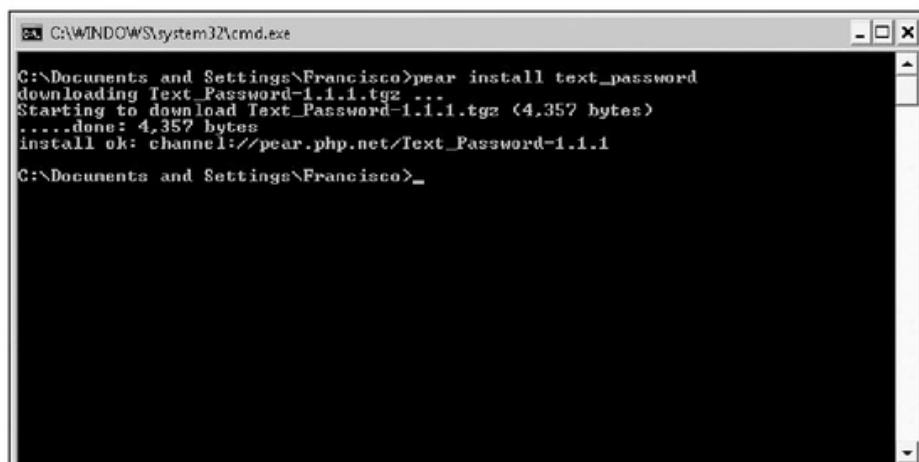


Figura 6. El paquete **Text_CAPTCHA** está disponible a través de PEAR, el repositorio de clases de PHP.

Para funcionar, necesita de PHP 4 o superiores y de la librería GD habilitada. Las dependencias de **Text_CAPTCHA** son:

- **Text_Password**
- **Numbers_Words**
- **Text_Figlet**
- **Image_Text**

Para instalar el paquete, se puede utilizar el comando **install** de PEAR:

```
C:\> pear install Text_Password
C:\> pear install Numbers_Words
C:\> pear install Text_Figlet
```

```
C:\> pear install Image_Text  
C:\> pear install Text_CAPTCHA
```

Recordemos siempre tener habilitada la librería GD2 en el archivo **php.ini**:

```
extension=php_gd2.dll
```

El paquete **Text_Password** genera frases aleatorias para generación de contraseñas mientras que **Image_Text** (requiere el soporte para TTF compilado y habilitado en PHP) crea una imagen a partir de una cadena de caracteres.

La clase **Text_CAPTCHA** provee unas pocas opciones y se caracteriza por su simplicidad. Para utilizarla, tendremos que incluir el archivo que contiene la clase principal:

```
require_once 'Text/CAPTCHA.php';
```

Y luego crear una instancia en la forma:

```
$captcha = Text_CAPTCHA::factory('Image');
```

El método **init** recibe cuatro argumentos: el ancho y el alto de la imagen en pixeles (los valores por defecto son 200px de ancho por 80px de alto), la frase a partir de la cual se generará la imagen, y una serie de opciones:

```
$captcha->init(150, 150, 'texto', $opciones);
```

Ninguno de estos argumentos es obligatorio. En cuanto a las opciones, son las disponibles para los objetos tipo **Image_Text**. Algunas de ellas:

- **text_color** (una cadena HTML o un array de colores RGB)
- **lines_color** (color de las líneas transversales)
- **background_color** (color de fondo)
- **font_path** (ruta al directorio donde se encuentra la fuente por utilizar, por ejemplo **C:\WINDOWS\Fonts**)
- **font_file** (archivo de la fuente, por ejemplo **cour.ttf**)
- **font_size** (tamaño de la fuente)

Podemos encontrar más información en http://pear.php.net/package/Image_Text.

```
$opciones['font_size'] = '14';
$opciones['font_path'] = './fuentes/';
$opciones['font_file'] = 'GARA.TTF';
$opciones['image_type'] = '#FF3300';

$captcha->init(150, 150, 'texto', $opciones);
```

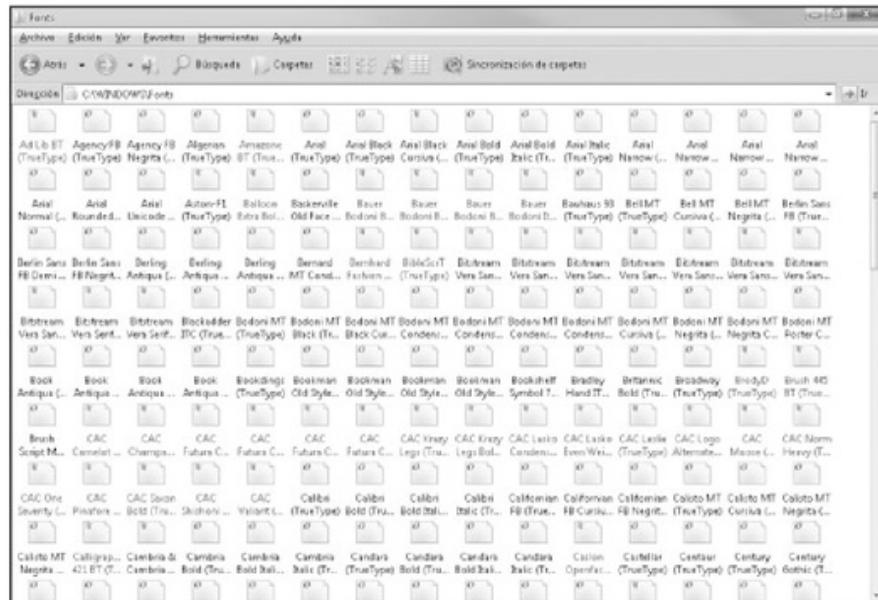


Figura 7. La implementación CAPTCHA de PEAR requiere que las rutas a las fuentes estén definidas.

Para generar las imágenes existen dos métodos: **getCAPTCHAAsJPEG** (devuelve una imagen en formato **JPG**) y **getCAPTCHAAsPNG** (devuelve una imagen en formato **PNG**):

```
$imagen = $captcha->getCAPTCHAAsPNG();
file_put_contents('./tmp/captcha.png', $imagen);
```

III FUNCIONES

Es importante destacar que las funciones **CType** (en el sitio oficial de PHP podemos obtener más datos acerca de la extensión que las agrupa: en la dirección web www.php.net/manual/ref ctype.php), **basename**, o **file_exists** pueden ser de utilidad para validar la consistencia, el origen y la existencia de los archivos invocados.

Una vez obtenida, se puede referenciar en un formulario, como por ejemplo:

```
<img src='./tmp/captcha.png' border="0" alt="captcha" />
Ingrese texto de la imagen: <input type="text" id="texto" name="texto" />
```

Para comparar lo que el usuario ingresó con lo que contiene la imagen, antes hay que guardar el texto, que se obtiene a través del método **getPhrase**:

```
$_SESSION['textoImagen'] = $captcha->getPhrase();
```

En el siguiente ejemplo completo, validamos el ingreso de datos mediante CAPTCHA a través de la implementación de **Text_CAPTCHA**:

- **index.php**

```
<?php

session_start();

if (isset($_POST['enviar'])) {
    if ($_SESSION['textoImagen'] == $_POST['texto']) {
        echo '<li>correcto';
        exit;
    } else {
        echo '<li>incorrecto';
    }
}

?>

<form method="post">
<table border="1" cellspacing="0" cellpadding="5" align="center">
    <tr>
        <td>Ingrese su nombre</td>
        <td><input type="text" id="nombre" name="nombre" /></td>
    </tr>
    <tr>
        <td>Ingrese su apellido</td>
```

```

<td><input type="text" id="apellido" name="apellido" /></td>
</tr>
<tr>
    <td>Ingrese texto de la imagen</td>
    <td>

        <img src='captcha.php' />
        <br />
        <input type="text" id="texto" name="texto" />

    </td>
</tr>
<tr>
    <td colspan="2" align="center">
        <input type="submit" id="enviar" name="enviar" />
        <input type="reset" />
    </td>
</tr>
</table>
</form>

```



Figura 8. CAPTCHA previene la automatización en el envío de formularios, aunque su uso no está ligado sólo a esa función.



SITIO ESPECIALIZADO

Es muy importante mencionar que el lenguaje de programación PHP mantiene un sitio dedicado al tema, en donde podremos encontrar recursos y artículos para comenzar a delinejar un plan de seguridad acorde con nuestras necesidades: <http://phpsec.org/library>. Allí se listan una serie de consejos prácticos para implementar en nuestros sistemas.

- **captcha.php**

```
<?php

session_start();

require_once 'PEAR/Text/CAPTCHA.php';

$captcha = Text_CAPTCHA::factory('Image');

$opcionesImg['font_size'] = '24';
$opcionesImg['font_path'] = './fuentes/';
$opcionesImg['font_file'] = 'cour.ttf';
$opcionesImg['text_color'] = '#660000';
$opcionesImg['lines_color'] = '#FF9966';
$opcionesImg['background_color'] = '#FFFFCC';

$captcha->init(200, 70, NULL, $opcionesImg);

$_SESSION['textoImagen'] = $captcha->getPhrase();

$imagen = $captcha->getCAPTCHAAsPNG();

if (PEAR::isError($imagen)) {
    printf('Error: %s',
        $imagen->getMessage());
    exit;
}

echo $imagen;
?>
```



FUNCIONES ONLINE

En el sitio web que se encuentra en la dirección <http://php-functions.com> se incluye una interfaz que nos permite ingresar una cadena de caracteres y recibir el resultado obtenido a partir de la aplicación de alguna de las funciones de encriptación de datos provistas por PHP. Puede sernos de utilidad para lograr resultados de manera rápida.

La imagen se genera directamente en **captcha.php**, pero podría almacenarse en un archivo y luego ser invocada desde el formulario, como vimos anteriormente.

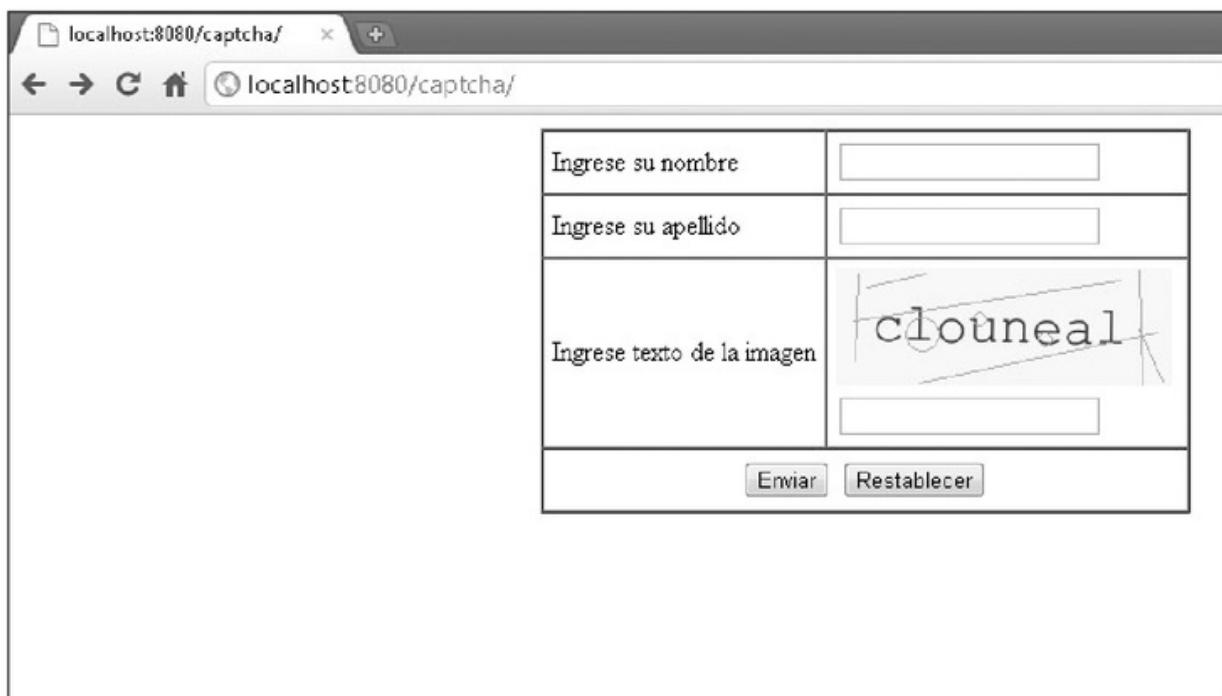


Figura 9. La generación dinámica de imágenes a partir de la biblioteca GD permite la implementación de soluciones CAPTCHA.

Más información en http://pear.php.net/package/Text_CAPTCHA.

ALMACENAR INFORMACIÓN EN BASE DE DATOS

Ya sea por cuestiones de comodidad o por inexperiencia, cierta información crítica es almacenada en bases de datos en formato de texto plano, sin ninguna clase de encriptación. Si la base fuera eventualmente accedida por terceros, el daño podría ser menor si, por ejemplo, las contraseñas de los usuarios no estuvieran directamente accesibles a simple vista, cosa que podemos lograr con un pequeño esfuerzo.

Un **hash** (resumen criptográfico) es algo así como un identificador único de un conjunto de datos (no puede haber un mismo hash para dos datos diferentes, o por lo menos es muy improbable), con la particularidad de que sólo convierte en un sentido: obtener el dato original a partir de un hash es poco menos que imposible.

Volviendo al ejemplo de las contraseñas, si alguien pudiera visualizar los hashes almacenados, no podría obtener a partir de ellos los textos originales. Para validar que la contraseña ingresada por el usuario exista en la base, habrá que obtener el hash del texto ingresado (a través de un formulario, por ejemplo) y compararlo con cada hash almacenado, asegurándonos que son exactamente idénticos.

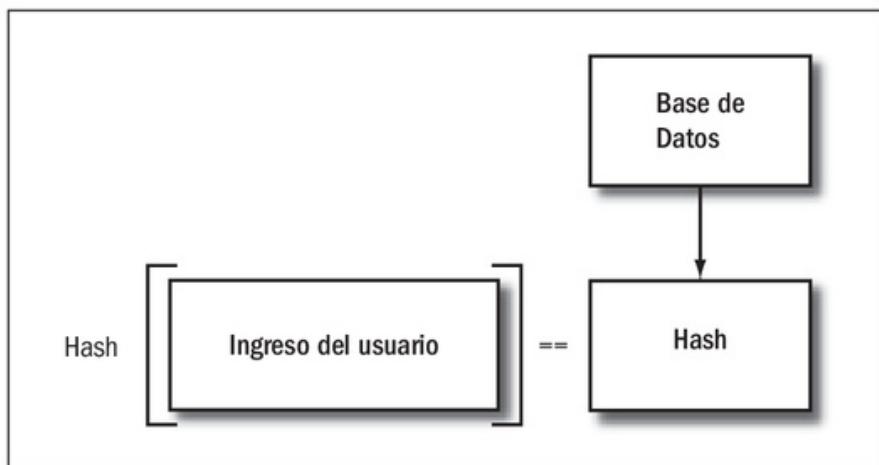


Figura 10. Típicamente, el hash almacenado se compara con la entrada del usuario.

PHP pone a sencilla disposición del programador una serie de poderosos algoritmos para encriptar información y, entre los más utilizados hoy en día, se encuentran SHA-1 (función **sha1**) y MD5 (función **md5**).

SHA-1 (*Secure Hashing Algorithm 1*) devuelve hashes de 160 bits (40 caracteres hexadecimales). Ésta es la implementación en PHP, pero existen especificaciones más fuertes, como SHA-256 y SHA-512 que brindan otros niveles de seguridad. Podemos encontrar más información en www.faqs.org/rfcs/rfc3174.



Figura 11. SHA-1 forma parte de los algoritmos más utilizados en PHP para la encriptación de datos.

El algoritmo **MD5**, por su parte, devuelve hashes de 128 bits (32 caracteres hexadecimales). Dentro del ámbito de una aplicación, ambos trabajan de manera similar. Podemos obtener más información en www.faqs.org/rfcs/rfc1321.

```
<?php

$cadena = 'Hola Mundo';

echo '<li><b>Cadena:</b> ' . $cadena;
echo '<li><b>SHA-1:</b> ' . sha1($cadena);
echo '<li><b>MD5:</b> ' . md5($cadena);

?>
```

Desde PHP 5.0.0, la función **sha1** recibe un segundo argumento opcional que, puesto a verdadero, devuelve el hash en formato binario puro de 20 posiciones, en lugar de número hexadecimal (comportamiento por defecto). Además, existe la función **sha1_file**, que devuelve el resumen criptográfico de un archivo (primer argumento). También admite el segundo argumento de la función **sha1**. La función **md5_file** tiene un comportamiento similar a **sha1_file**.

Por su parte, **md5** también admite un modo binario: si el valor del segundo parámetro es verdadero, el resumen **md5** devuelto tiene una longitud de 16 bits, y su formato es binario. Veremos más de **md5** en los próximos ejemplos.

Un formulario de registro podría ser:

```
<?php

if (isset($_POST['enviar'])) {
    //... conexion ...
    //... validacion ...

    $nombreUsuario = $_POST['nombre'];
    $passwordUsuario = sha1($_POST['oPassword']);

    $res = mysql_query("insert into usuarios (nombreUsuario, passwordUsuario)
        values ('$nombreUsuario', '$passwordUsuario')");
    if ($res) {
        //... ingresado ...
    } else {
        //... no ingresado ...
    }
}
```

```
//... redireccion ...  
}  
  
?>  
  
<form method="post" action="index.php">  
<table>  
  <tr>  
    <td>Ingrese Su Nombre</td>  
    <td><input type="text" id="nombre" name="nombre"></td>  
  </tr>  
  <tr>  
    <td>Ingrese Password</td>  
    <td><input type="password" id="oPassword" name="oPassword"></td>  
  </tr>  
  <tr>  
    <td>Repita Password</td>  
    <td><input type="password" id="rPassword" name="rPassword"></td>  
  </tr>  
  <tr>  
    <td colspan="2"><input type="submit" id="enviar" name="enviar"  
           value="Enviar Formulario"></td>  
  </tr>  
</table>  
</form>
```

Si el usuario ingresa como contraseña **mipassword**, el resultado se vería:

```
c2cd851a3c66015bba94c64f443f510a05ec7365
```

Y cuando quiera ingresar en el sitio:

```
<?php  
  
if (isset($_POST['enviar'])) {  
  //... conexion ...  
  //... validacion ...
```

```

$nombreUsuario = $_POST['nombre'];
$passwordUsuario = sha1($_POST['oPassword']);

$res = mysql_query("select * from usuarios where nombreUsuario =
    '$nombreUsuario' and passwordUsuario = '$passwordUsuario'");
if (mysql_num_rows($res)) {
    //... correcto ...
} else {
    //... incorrecto ...
}

//... redireccion ...
}

?>

<form method="post" action="index.php">
<table>
<tr>
    <td>Ingrese Su Nombre</td>
    <td><input type="text" id="nombre" name="nombre"></td>
</tr>
<tr>
    <td>Ingrese Password</td>
    <td><input type="password" id="oPassword" name="oPassword"></td>
</tr>
<tr>
    <td colspan="2"><input type="submit" id="enviar" name="enviar"
        value="Enviar Formulario"></td>
</tr>
</table>
</form>

```

Esta clase de algoritmos pueden ser violados a través de scripts que generan cadenas de caracteres comunes, tomadas tal vez de un diccionario, o contraseñas habitualmente utilizadas, para luego obtener el hash de cada una de ellas y compararlos con los almacenados en la base de datos (ataque por fuerza bruta). Por eso, algunos sitios obligan al usuario a incorporar caracteres numéricos en sus contraseñas, a no incluir el nombre de usuario en ellas y a que, además, tengan una longitud determinada (más de n caracteres).

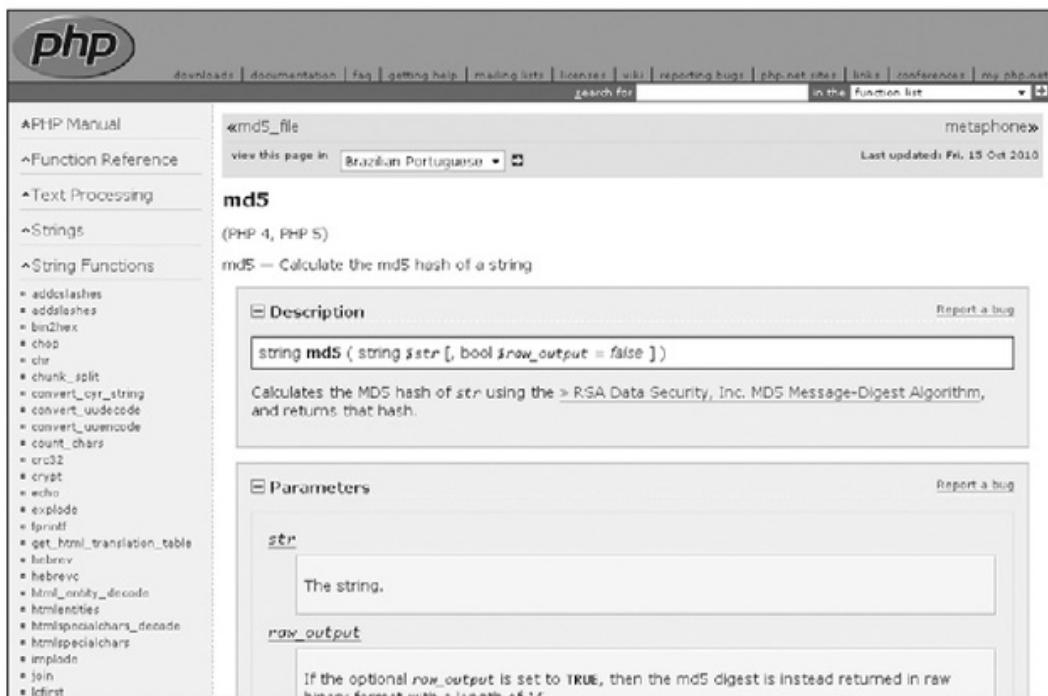


Figura 12. PHP provee implementaciones de múltiples algoritmos para la encriptación de datos en aplicaciones.

El sitio <http://php-functions.com> incluye una interfaz para ingresar una cadena de caracteres y recibir el resultado obtenido a partir de la aplicación de alguna de las funciones de encriptación de datos provistas por PHP.

Otra función disponible es `crypt` que, al igual que las anteriores, utiliza algoritmos no reversibles. Aplica el método estándar de encriptación en Unix, llamado DES, y los argumentos que recibe son la cadena por encriptar y un string en el cual se basará la encriptación. El tamaño y el comienzo de este argumento dependerán del tipo de encriptación por utilizar, como veremos a continuación.

Existen diferentes tipos de encriptación que dependen del sistema y que se refieren a través de las constantes (1 si está habilitado, 0 si no):

- **CRYPT_STD_DES** (DES estándar de 2 caracteres).
- **CRYPT_EXT_DES** (DES extendida de 9 caracteres).
- **CRYPT_MD5** (MD5 de 12 caracteres comenzando por **\$1\$**).
- **CRYPT_BLOWFISH** (DES extendida de 16 caracteres comenzando por **\$2\$** o **\$2a\$**).

```
<?php

if (CRYPT_STD_DES) {
$cadena1 = crypt('texto', 'xx');
$cadena2 = crypt('texto', 'fd');
```

```
$cadenaN = crypt('texto', 'nn');

}

if (CRYPT_BLOWFISH) {
$cadena1 = crypt('texto', '$2a$abcdefhijkl');

}

?>
```

La función toma los primeros 8 caracteres del primer argumento. Esto significa que, si se encriptan dos cadenas distintas cuyos primeros 8 caracteres coinciden y, además, se utiliza el mismo segundo argumento, la salida será la misma.

La extensión **mcrypt** pone a disposición de los desarrolladores de PHP una serie de poderosos algoritmos de encriptación de datos, que pueden obtenerse mediante la función **mcrypt_list_algorithms**, como muestra el siguiente ejemplo:

```
<?php

echo '<pre>';
echo print_r(mcrypt_list_algorithms(), TRUE);
echo '</pre>';

/*
Array
(
    [0] => cast-128
    [1] => gost
    [2] => rijndael-128
    [3] => twofish
    [4] => arcfour
    [5] => cast-256
    [6] => loki97
    [7] => rijndael-192
    [8] => saferplus
    [9] => wake
```

```
[10] => blowfish-compat
[11] => des
[12] => rijndael-256
[13] => serpent
[14] => xtea
[15] => blowfish
[16] => enigma
[17] => rc2
[18] => tripledes
)
*/
?>
```

Hay dos funciones principales: **mcrypt_encrypt** y **mcrypt_decrypt**, para encriptar y desencriptar cadenas. La primera recibe cinco argumentos:

- Algoritmo
- Clave
- Texto por encriptar
- Modo (precedido por el prefijo **MCRYPT_MODE_** + el modo en mayúsculas).
- Iv

La clave es una cadena de caracteres que permitirá luego desencriptar la información. Representa un valor importante y deberá permanecer en un lugar seguro.

En cuanto al modo, existe uno para cada tipo de encriptación que se quiera llevar a cabo (encontraremos más información en <http://php.net/mcrypt>). También podemos obtener un listado a través de la función **mcrypt_list_modes**:

```
<?php
echo '<pre>';
echo print_r(mcrypt_list_modes(), TRUE);
echo '</pre>';

/*
Array
```

```

(
    [0] => cbc
    [1] => cfb
    [2] => ctr
    [3] => ecb
    [4] => ncfb
    [5] => nofb
    [6] => ofb
    [7] => stream
)
*/
?>

```

El último argumento está relacionado con el modo y el algoritmo seleccionados, y permite inicializar el proceso de encriptación. En el siguiente código podemos ver un ejemplo de uso de la librería con los parámetros establecidos.

```

<?php

$clave = "1234";
$cadena = "este es el texto";
$algoritmo = "rijndael-128";
$modo = MCRYPT_MODE_ECB;

$iv = mcrypt_create_iv(mcrypt_get_iv_size($algoritmo, $modo), MCRYPT_RAND);

$cadenaEncriptada = mcrypt_encrypt($algoritmo, $clave, $cadena, $modo, $iv);

$cadenaDesencriptada = mcrypt_decrypt($algoritmo, $clave,
$cadenaEncriptada, $modo, $iv);

echo '<li>'.$cadena;
echo '<li>'.$cadenaEncriptada;
echo '<li>'.$cadenaDesencriptada;

?>

```

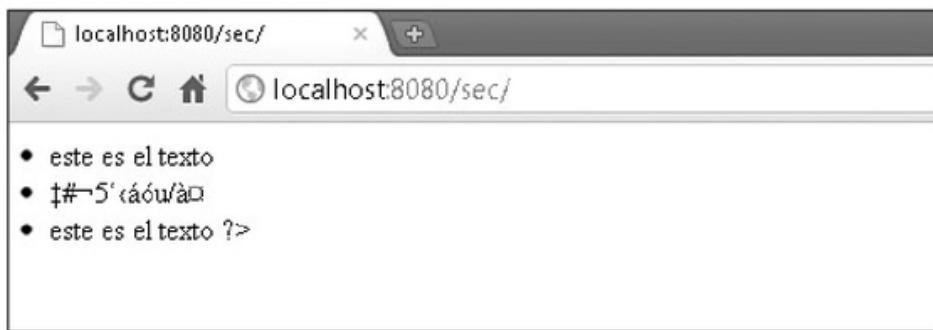


Figura 13. La extensión mcrypt es una de las más populares y versátiles opciones en cuanto a la encriptación de datos utilizando claves públicas.

Esta extensión es una implementación de la biblioteca del mismo nombre. En caso de no estar disponible, podemos descargarla desde: <http://mcrypt.sourceforge.net>.

Recordemos que, para habilitar la extensión, la línea correspondiente no deberá estar comentada en el archivo **php.ini**:

```
extension=php_mcrypt.dll
```

En sistemas Unix, además, hay que compilar PHP con la opción **—with-mcrypt**. Podemos obtener más información en <http://php.net/mcrypt>.

PEAR provee paquetes relacionados con la criptografía en aplicaciones web (http://pear.php.net/package/Crypt_RSA).

Estas técnicas tienen como contra la imposibilidad de recordarles contraseñas a los usuarios. Es común ver sitios que admiten la opción *¿Olvidó su password? Haga clic aquí*, que típicamente envía la clave al e-mail del usuario. Una solución podría ser la posibilidad de ingresar una nueva contraseña si se responde a una pregunta de seguridad, accediendo a través de un enlace enviado a una dirección de correo electrónico, por ejemplo. Según estudios realizados, el algoritmo MD5 ha probado no ser del todo seguro. Para complejizar la tarea de revertir la cadena encriptada, es posible incluir prefijos propios de la aplicación, por ejemplo:

```
<?php  
  
$password = md5('abcdef'.md5($_POST['password']));  
  
?>
```

El prefijo deberá almacenarse para luego poder realizar el proceso inverso.

El caso de MySQL

MySQL por su lado, posee implementaciones propias de estos algoritmos disponibles a través de las funciones **md5** (desde MySQL versión 3.23.2) y **sha1** (desde MySQL versión 4.0.2), accesibles de la siguiente manera:

```
mysql> select MD5('esta es la cadena a encriptar');
mysql> select SHA1('esta es la cadena a encriptar');
```

O desde PHP:

```
$res = mysql_query("insert into usuarios (nombreUsuario, passwordUsuario)
values ('$nombreUsuario', SHA1('$POST[oPassword]'))");
```

Además, posee su propia función para encriptar información: **password**.

```
mysql> select PASSWORD('esta es la cadena a encriptar');
```

The screenshot shows the MySQL Documentation website. The top navigation bar includes links for Developer Zone, Downloads, Documentation (which is highlighted), MySQL Manual Archives, MySQL Workbench, Expert Guides, Topic Guides, MySQL Cluster, Other Docs, MySQL Wiki, and About. Below the navigation, there's a search bar and a login/register link. The main content area displays the MySQL 5.1 Reference Manual, specifically the section on Password Security. It discusses how user accounts are listed in the `user` table and how each account is assigned a password. It also explains that the password stored in the `Password` column is not the plain-text version but a hash value computed from it. The `PASSWORD()` function is mentioned as being used in two phases of client/server communication: authentication and privilege management. A sidebar on the right lists other sections under "5.3.2.3. Password Hashing in MySQL". At the bottom of the page, there's a "Section Navigation" sidebar with links to various MySQL security and hashing topics.

Figura 14. MySQL provee sus propias funciones para implementar algoritmos de encriptación en instrucciones SQL.

Podemos obtener más información acerca de éstas y otras funciones de encriptación de datos con diversos algoritmos sobre MySQL en el sitio oficial del motor: www.mysql.com, o en el manual de la distribución oficial.

Seguridad en bases de datos

A pesar de que las bases de datos poseen sus propios sistemas de seguridad y aceptan limitar los permisos de los usuarios en lo referido al acceso y tratamiento de la información almacenada, los scripts PHP que acceden a sus servicios pueden llegar a ser una puerta de entrada en caso de no tomar las precauciones necesarias.

Un caso para tener en cuenta es el de los datos de conexión (nombre de usuario, contraseña, nombre del servidor, puerto, driver del motor, nombre de la base de datos, etcétera). Normalmente, una misma conexión es utilizada desde múltiples páginas, lo que supone centralizar esos datos en un mismo archivo que deberá estar ubicado por fuera del **DocumentRoot** (directiva del **httpd.conf** que define el directorio raíz del sitio). De esta forma, no se podrá acceder a él directamente a través de un navegador (esto es importante si el archivo posee una extensión diferente de **.php**, por ejemplo **.inc** o **.xml**, que es tratada por el servidor web como texto plano). Luego, mediante funciones tales como **include**, **include_once**, **require**, o **require_once** será posible disponer de la conexión.

Esto vale también para aquellas bases orientadas a archivos (y en general para cualquier documento no público), como SQLite o MS Access, por ejemplo. En estos casos, sería recomendable ubicar las bases por fuera del directorio raíz del sitio.

Otro método de acceso a una base de datos es a través del llamado **SQL injection**, que consiste en incluir en las partes variables de una instrucción SQL valores que la modifiquen. El acceso a estas variables se produce por medio de formularios.

Supongamos un formulario para que los usuarios registrados ingresen en un sistema introduciendo su nombre de usuario y contraseña:

```
<form method="POST">
<table>
<tr>
    <td>Nombre de usuario:</td>
    <td><input type="text" id="username" name="username" /></td>
</tr>
<tr>
    <td>Password:</td>
    <td><input type="password" id="password" name="password" /></td>
</tr>
<tr>
    <td colspan="2"><input type="submit" value="enviar" /></td>
</tr>
</table>
</form>
```

En la página destino, suponiendo que trabajemos con la extensión **mysql**, hallaremos:

```
<?php

include '/config/conexion.inc';

$res = mysql_query("select * from usuarios where username = '$_POST
[username]' and password = '$_POST[password]');

if (mysql_num_rows($res)) {
    //correcto
} else {
    //error
}

?>
```

Ésta es una consulta típica, un usuario podría llegar a intuirlo. Si además completa el primer campo del formulario con algo como:

```
'xxx' or 1 = 1 -
```

La sentencia SQL final quedaría:

```
select * from usuarios where username = 'xxx' or 1 = 1 -' and password = ''
```

La cadena — indica el comienzo de un comentario en MySQL, por lo tanto, lo que se evalúa finalmente por el motor es:

```
select * from usuarios where username = 'xxx' or 1 = 1
```

La segunda condición siempre se cumple, por lo que siempre se ingresa en el sistema. Hay muchísimas variantes de este tipo de ataques, que pueden neutralizarse filtrando los datos variables de la consulta SQL a través de la función **mysql_real_escape_string**:

```
<?php
```

```

include '/config/conexion.inc';

$username = mysql_real_escape_string($_POST[username]);
$password = mysql_real_escape_string($_POST[password]);

$res = mysql_query("select * from usuarios where username = '$username'
    and password = '$password'");

if (mysql_num_rows($res)) {
    //correcto
} else {
    //error
}

?>

```

Esta función está disponible sólo para **mysql**, pero todas las demás normalmente cuentan con funciones para escapar cadenas antes de utilizarlas en una instrucción SQL.

```
select * from usuarios where username = 'xxx\' or 1 = 1 -' and password = ''
```

Incluso, se podrían utilizar funciones especiales de PHP para este fin como **addslashes** o teniendo la directiva **magic_quotes_gpc** habilitada (que de forma automática llama a la función **addslashes** para todos los datos que llegan a través de los métodos **GET** y **POST**, y a través de las cookies).



Figura 15. El valor de la directiva **magic_quotes_gpc** influye en el comportamiento de la función **addslashes**.

La función **get_magic_quotes_gpc** obtiene el valor actual de **magic_quotes_gpc**.

En el caso de MySQL en particular y los gestores de bases de datos en general, las funciones disponibles a través de las distintas extensiones de PHP para escapar datos antes de introducirlos en una instrucción SQL no reemplazan determinados caracteres como % (porcentual) y _ (guión bajo). Esto sucede porque dentro del lenguaje SQL hay ciertos comandos que se valen de estos signos para acceder y manipular los componentes de la base de datos. Como ejemplo de estos comandos, y tomando a MySQL como base, podemos citar a **LIKE**, **GRANT**, y **REVOKE**. Podemos obtener más información acerca de la sintaxis de estas construcciones en el manual oficial de MySQL, más precisamente en <http://dev.mysql.com/doc/refman/5.1/en/index.html>.

INCLUSIÓN DE ARCHIVOS

Es muy común incluir archivos externos dentro de una aplicación web, por ejemplo, para modularizar las tareas, reutilizar fragmentos de código o recuperar datos de configuración. PHP brinda cuatro funciones (en realidad, construcciones) al respecto: **include**, **include_once**, **require**, y **require_once**. Todas tienen un comportamiento similar que consiste en recibir la ruta hacia un archivo como argumento para luego incluirlo en el lugar desde donde se realizó la llamada y evaluar su contenido. La diferencia entre **include** y **require** es que, si el archivo no se encuentra, la primera produce una simple advertencia mientras que la segunda, un error fatal.

El contenido de los archivos incluidos puede disponer de las variables previamente utilizadas y modificar sus valores. A diferencia de JavaScript, si el archivo contiene código PHP, éste deberá estar encerrado entre las etiquetas de apertura y cierre.

Dentro del archivo **php.ini**, se definen dos directivas relacionadas: si **allow_url_fopen** está habilitada, se podrán incluir archivos a través de URLs. La directiva **include_path**, por su parte, define una lista de directorios en donde las funciones buscarán archivos en caso de no encontrarlos en el directorio actual. Es recomendable mantener deshabilitadas las directivas **allow_url_fopen** y **allow_url_include**, utilizando las funciones **CURL** (*Client URL Library*) como reemplazo.

Como regla general, se suele decir que todos los archivos no públicos (aquellos que no tienen por qué ser accesibles vía URL) deben ubicarse por fuera del **DocumentRoot** (directiva del **httpd.conf** que define el directorio raíz del sitio), sobre todo si la extensión no es reconocida por el servidor web como código PHP.

El SQL injection es similar a lo que podría llegar a suceder con la inclusión dinámica de archivos por URL o por cualquier otro tipo de envío:

```
<?php include "/inc/$_GET[nombreArchivo]"; ?>
```

Este método es conocido como **Code injection** y, si no validamos correctamente los datos que ingresan en la aplicación, el acceso a directorios o archivos del servidor podría ser inevitable. Otro punto para tener en cuenta es la posibilidad de que los archivos incluidos en un script puedan llegar a ser accedidos directamente, por lo que tendremos que aplicar la misma protección en ambos.

AJAX

Esta clase de aplicaciones se caracterizan por diferir del modo de funcionamiento tradicional, en especial en cómo se conecta un cliente (un navegador) con un servidor. Habitualmente, las peticiones interrumpen la recepción de datos del cliente, quien debe esperar a que la página se actualice con la nueva información recuperada. El modelo **Ajax** introduce un intermediario entre el cliente (navegador) y el servidor, o si se quiere divide al cliente en dos partes: la presentación (interfaz de usuario) y el motor Ajax (aplicación escrita en JavaScript).

La interfaz de usuario se comunica con el motor Ajax mediante JavaScript, y éste envía una petición en background (segundo plano) al servidor mediante el objeto **XMLHttpRequest** (en el **Capítulo 3** vimos más información acerca de él). Cuando el servidor completa su procesamiento, devuelve la respuesta (en formato XML o JSON, por ejemplo) al motor Ajax, que a su vez actualiza datos en la interfaz, la cual se mantiene durante todo el proceso a la vista del usuario. La actualización se realiza a través de DHTML y DOM sobre elementos HTML como DIVs o SPANs, por ejemplo.

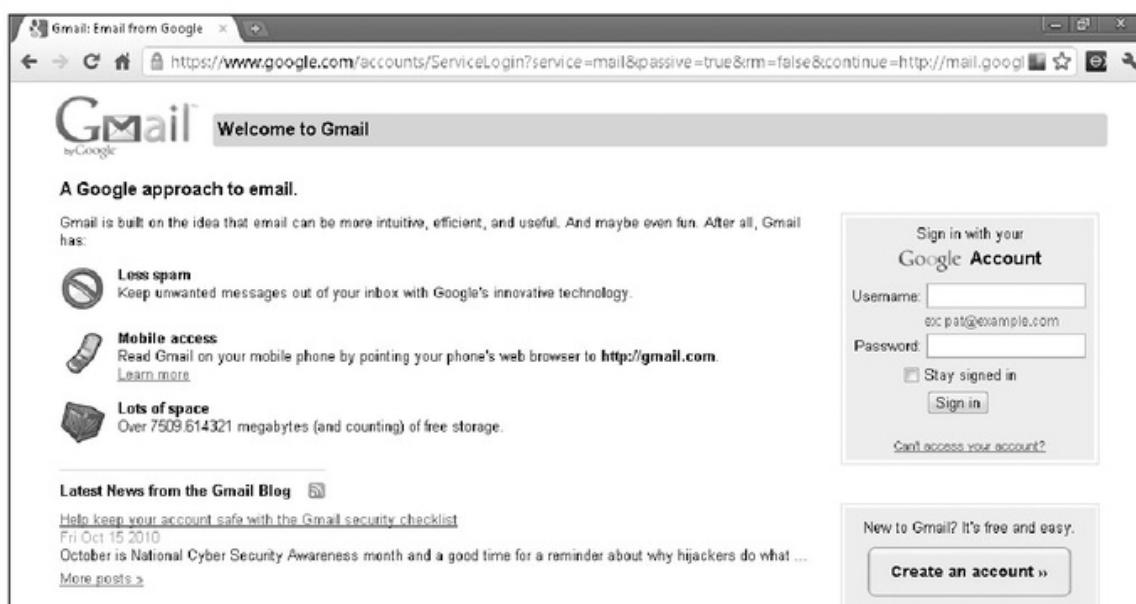


Figura 16. Las aplicaciones Ajax precisan, por sus características, un particular cuidado en relación con el acceso a la información.

Ahora bien, acceder a datos almacenados en un servidor desde una aplicación cliente (JavaScript) supone un control extra acerca de quién accede y cómo lo hace. Una de las necesidades en cuanto a la seguridad será validar el acceso a los documentos que retornan datos. Esta validación puede implementarse a partir de sesiones para identificar al cliente y, desde ese dato, permitir o restringir los accesos a los datos almacenados. Otro punto es la habilitación de JavaScript, un lenguaje que tiene un papel muy importante en Ajax.

La mayoría de los navegadores actuales lo soporta, pero pueden suceder tres cosas:

- Que los usuarios no utilicen navegadores actuales.
- Que el soporte esté deshabilitado (total o parcialmente).
- Que el soporte para objetos ActiveX esté deshabilitado (necesario para algunas versiones de Internet Explorer).

Esto podría ir en contra del correcto funcionamiento de las aplicaciones Ajax y generar agujeros de seguridad o acceso a información a través de mensajes de error, por ejemplo. La solución es tener en cuenta las cualidades posibles del cliente y actuar en consecuencia.

LÍNEA DE COMANDOS

PHP es considerado un lenguaje de propósito general, más allá de que su principal utilización se dé en sitios web. Entre las múltiples funciones ofrecidas en la versión base, se encuentran las relacionadas con el sistema de archivos del sistema operativo:

- **exec**
- **system**
- **passthru**
- **shell_exec**

La función **exec** permite ejecutar comandos del sistema operativo. Recibe tres argumentos: el comando por ejecutar, un array que contendrá la salida del comando y una variable que almacenará el resultado devuelto (**true**, **false**, un escalar, etcétera). Por su parte, la función **system** ejecuta un programa externo recibiendo el comando como primer argumento. En el segundo, recupera el resultado obtenido. A través de **passthru**, es posible ejecutar un programa externo y guardar la salida pura en la variable pasada como segundo argumento. Por último, la función **shell_exec** ejecuta un comando y devuelve la salida correspondiente. No está habilitada en modo seguro (**safe_mode**). Además, PHP incorpora una serie de funciones (**popen**, **proc_open**, **proc_get_status**,

proc_nice, proc_close, proc_terminate) que permiten ejecutar comandos e interactuar con las aplicaciones. Nuevamente, si los argumentos a estas funciones pueden ser definidos de una manera u otra por un usuario, el sistema corre un alto riesgo. Existen funciones específicas para escapar comandos pasados a esta clase de funciones, como por ejemplo, **escapeshellcmd**, que recibe como argumento el comando y lo devuelve normalizado. Una alternativa similar es **escapeshellarg**.

El modo seguro restringe la operación de estas funciones: en caso de estar activado, solamente se podrán ejecutar los programas que se encuentren listados en la directiva **safe_mode_exec_dir** del archivo **php.ini**.

REGISTER GLOBALS

A pesar de que la directiva **register_globals** está deshabilitada por defecto (4.2.0 y superiores) e, incluso, las últimas versiones de PHP ya no la incorporan, hay que tener presente los inconvenientes que pueden llegar a producirse cuando, por algún motivo, el servidor donde se alojan las páginas la mantiene activa.

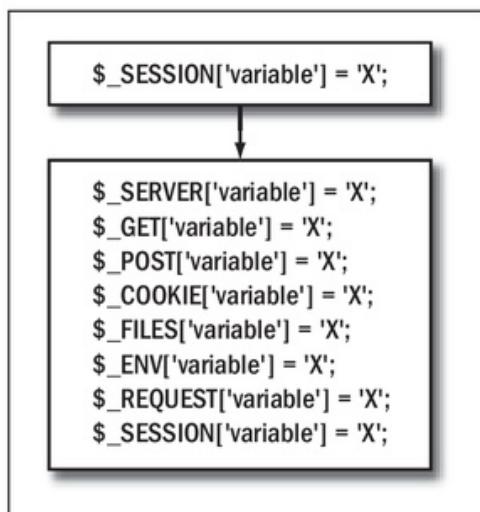


Figura 17. La directiva **register_globals** ya no está habilitada en PHP por razones de seguridad.

Con **register_globals**, se pone de manifiesto uno de los puntos más importantes en lo referido al filtrado de datos: verificar su origen. Permitía, en caso de estar habilitada, asumir todas las variables como globales, sin importar de qué tipo eran (de sesión, pasadas por formulario, locales de un script, etcétera). Este comportamiento generó inconvenientes en la seguridad en PHP3 y, ya en las versiones 4 y 5, fue posible configurarlo desde el archivo **php.ini**. En PHP6, directamente no está disponible: no se registran variables como globales. Las aplicaciones que trabajan con **register_globals = On** deberán revisarse y, eventualmente, ser modificadas.

Para diversificar los distintos tipos de variables, PHP utiliza diferentes arrays (matrices predefinidas a partir de la versión 4.1.0):

- **`$_SERVER`** (variables del servidor).
- **`$_GET`** (variables recibidas a través del método **GET**).
- **`$_POST`** (variables recibidas a través del método **POST**).
- **`$_COOKIE`** (cookies).
- **`$_FILES`** (archivos).
- **`$_ENV`** (variables de entorno).
- **`$_REQUEST`** (variables recibidas a través de los métodos **GET** y **POST**).
- **`$_SESSION`** (variables de sesión).

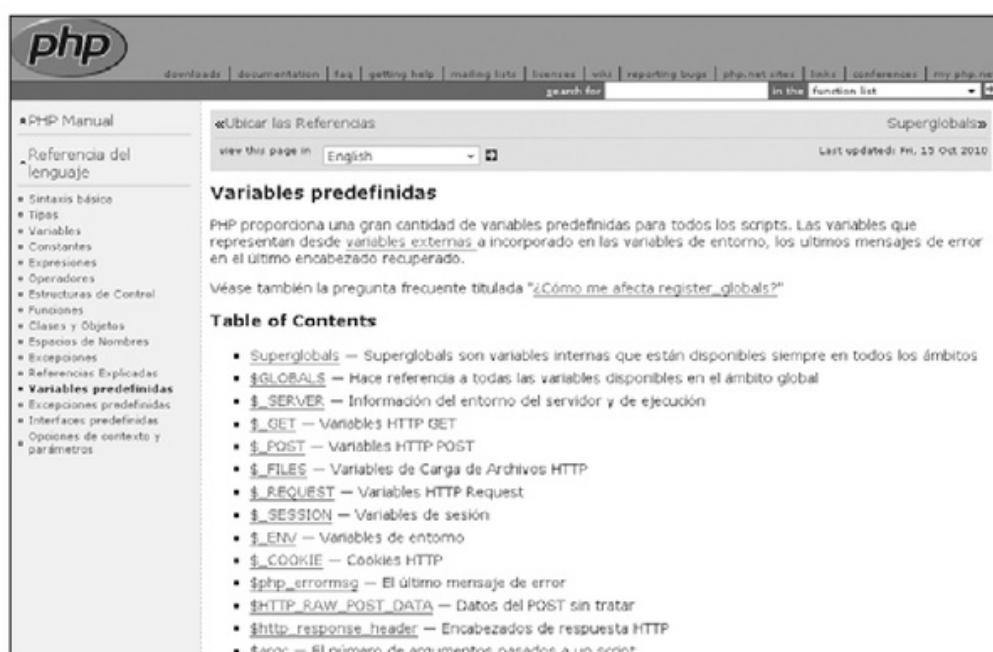


Figura 18. En las últimas versiones, PHP modificó el acceso a variables predefinidas a través de arrays especiales.

Supongamos un script para autenticar usuarios. Luego de un determinado proceso de ingreso y validación de datos, si el usuario se identifica correctamente, almacenamos en la variable de sesión **`$_SESSION[usuario]`** el valor registrado.

```
http://www.misitioweb.com?usuario=registrado
```

Con **register_globals** habilitada, se consideraría automáticamente autorizado, evitando el proceso de registro (PHP no requiere declaración de variables). Con **register_globals** deshabilitada, podríamos especificar la forma de recepción de la variable. Si definimos que la variable será de tipo sesión (accesible mediante **`$_SESSION`**), no nos importa el valor contenido en **`$_GET[usuario]`**, sino el de **`$_SESSION[usuario]`**.

Si **register_globals** se encuentra referenciada en algún archivo de configuración o dentro de una página, PHP6 emitirá un mensaje de error al respecto (**E_CORE_ERROR**).

Modo seguro en PHP

PHP puede trabajar en modo seguro (*safe mode*). Para activar este comportamiento, habrá que habilitar la directiva **safe_mode** del archivo **php.ini**. Hay múltiples directivas asociadas a **safe_mode** que tendrán efecto en el comportamiento de los scripts una vez que la habilitación se haya llevado a cabo:

- **safe_mode_exec_dir**: especifica el directorio en donde deberán residir los scripts. Cualquier llamada a un archivo **.php** que esté en el directorio no se llevará a cabo. Incluso, las llamadas a otros programas que no estén allí.
- **safe_mode_allowed_env_vars**: contiene un listado de prefijos que identifican las variables de entorno que el usuario podrá modificar. Cualquier variable de entorno que no comience con ese prefijo permanecerá inalterable.
- **safe_mode_protected_env_vars**: contiene un listado de las variables de entorno que el usuario no podrá modificar, aun cuando la directiva **safe_mode_allowed_env_vars** diga lo contrario, esta lista las mantiene bloqueadas. Algunas de las variables de entorno disponibles en PHP son:

- **allusersprofile**
- **classpath**
- **commonprogramfiles**
- **computernamespace**
- **comspec**
- **fp_no_host_check**
- **number_of_processors**
- **os**
- **path**
- **pathext**
- **processor_architecture**
- **processor_identifier**

III SAFE MODE

Si necesitamos investigar más sobre este tema podemos hallar más información sobre las características del modo seguro en <http://ar.php.net/manual/es/features.safe-mode.php>. Allí encontraremos los detalles acerca de las cualidades más importantes y cómo pueden llegar a afectar los scripts, incluyendo las directivas no disponibles.

- **processor_level**
- **processor_revision**
- **programfiles**
- **qtjava**
- **systemdrive**
- **systemroot**
- **temp**
- **tmp**
- **userprofile**
- **windir**
- **ap_parent_pid**

- **open_basedir**: permite definir una serie de directorios. Si se intenta abrir un archivo que no esté contenido en esa carpeta o por debajo de ella, PHP no permitirá el acceso. En caso de estar vacía, no habrá restricciones.
- **enable_dl**: permite la utilización o no de la función **dl**, que admite cargar módulos de manera dinámica. En *safe mode*, está deshabilitada más allá del valor de **enable_dl**.
- **disable_functions**: recibe un listado de funciones separadas por comas que PHP simplemente ignorará. Se utiliza frecuentemente para deshabilitar funciones como **mail**, **system**, **phpinfo**, etcétera; es decir, aquellas que puedan ser ejecutadas para sacar provecho de los recursos del servidor o para obtener información acerca de sus características por personas ajena al sistema.

Otro punto para tener en cuenta: PHP no permitirá el acceso a documentos que no estén por debajo de **doc_root** (directiva del **php.ini**).

Es importante aclarar que su uso está desaconsejado a partir de PHP 5.3.0. Podemos obtener más detalles acerca de cuáles opciones serán mantenidas y cuáles ya no serán tenidas en cuenta visitando el sitio web oficial del lenguaje, más precisamente, la dirección www.php.net/manual/es/ini.sect.safe-mode.php.

Conocer las características del modo seguro nos ayudará, sin embargo, a realizar la detección y posterior corrección de ciertos comportamientos no deseados en scripts que se ejecuten bajo versiones inferiores a la 5.3.0.

III SESIONES

Es importante recordar que uno de los posibles ataques al trabajar con sesiones es la usurpación de identidad, que consiste en definir a través de una URL un identificador de sesión. Si otro usuario está utilizando el mismo valor, ambos accederán con exactos privilegios, y podrán ocasionar daños considerables.

Manipulación del archivo php.ini

PHP se conduce de una forma o de otra en parte según los valores registrados en el archivo de configuración **php.ini**. Es posible conocer los valores de múltiples maneras, entre ellas, a través de la función **phpinfo**:

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

Con **ini_get** (recibe el nombre de una directiva y devuelve su valor actual):

```
<?php echo ini_get('display_errors'); ?>
```

Para modificar los valores, está disponible la función **ini_set** (recibe como argumentos el nombre de una directiva y el nuevo valor). No todas las directivas del archivo de configuración de PHP admiten esta opción.



Figura 19. La función **phpinfo** brinda información valiosa acerca del sistema y, en ocasiones, permanece deshabilitada por cuestiones de seguridad.

III CAPTURA DE SESIONES

Debemos tener en cuenta que la captura de sesiones puede llegar a ser un riesgo para una aplicación web. En estos casos, una forma de neutralizar esta clase de acciones maliciosas es a través del llamado **SSL** (*Socket Secure Layer*), que se encarga de permitir la encriptación de los datos intercambiados entre un cliente y un servidor.

SEGURIDAD EN PHP 6

Además de las modificaciones mencionadas, en la versión 6 se incluirán las funciones disponibles en Hardened PHP (<http://hardened-php.org>) que implementan opciones adicionales relacionadas con la seguridad en scripts, por ejemplo:

- El **header** no podrá recibir más de una cabecera a la vez.
- La directiva **allow_url_fopen** se dividirá en **allow_url_fopen** para **fopen** y **allow_url_include** para **include**.



Figura 20. Los sitios seguros son valorados tanto por las empresas responsables como por los usuarios que acceden a sus servicios.



RESUMEN

La seguridad en aplicaciones web es un tema que no se puede obviar en estos tiempos que corren, y PHP pone a disposición de sus desarrolladores muchas herramientas que permiten incorporar niveles de seguridad más que decentes para toda clase de proyectos. Ya sea a través de extensiones como de clases externas, las opciones son variadas y abarcan temas como filtrado de datos, formularios, base de datos y encriptación, entre otros.



ACTIVIDADES

PREGUNTAS TEÓRICAS

1 ¿Cree que las opciones puestas a disposición por PHP en cuanto a seguridad son suficientes?

2 ¿Cree que sus aplicaciones son seguras?

3 ¿Tomaría los mismos recaudos para aplicaciones que son accedidas por muchos usuarios que para las privadas?

4 ¿Qué riesgos potenciales conlleva la función include?

5 ¿Qué es el modo seguro?

6 ¿Qué es CAPTCHA? ¿En qué casos lo utilizaría?

7 ¿Qué riesgo extra genera una aplicación Ajax?

8 ¿Cuáles son las opciones para implementar encriptación?

9 ¿Qué es el filtrado de datos?

10 ¿Qué significa escapar una cadena?

EJERCICIOS PRÁCTICOS

1 Genere una aplicación que utilice CAPTCHA para validar el registro de usuarios.

2 Encripte contraseñas de usuarios antes de almacenarlas en una base de datos e implemente un login para validar el ingreso.

3 Busque información acerca de las diferencias entre las funciones mysql_escape_string y mysql_real_escape_string.

4 Busque información acerca del algoritmo SHA-1.

5 Busque información acerca del algoritmo MD5.

PEAR

El repositorio oficial de clases nos permite extender la potencia del lenguaje. Por eso, en este capítulo explicamos los pasos necesarios para instalar, configurar y utilizar PEAR.

También algunas de las opciones disponibles: MDB2, la popular capa de abstracción para acceder a bases de datos; Spreadsheet_Excel_Writer, para generar documentos XLS desde datos disponibles en distintos orígenes y HTML_QuickForm, para generar y validar formularios.

Instalación	266
Opciones disponibles	270
Utilización	272
Errores	273
Implementaciones	273
Acceso a datos con MDB2	273
Hojas de cálculo con Spreadsheet_Excel_Writer	290
Formularios	
con HTML_QuickForm	301
Agregar elementos	302
Validar formularios	305
Enviar datos	310
Salida sin tablas	311
Resumen	313
Actividades	314

INSTALACIÓN

La distribución oficial, a menos que hayamos definido explícitamente lo contrario durante la instalación, incorpora lo que se denomina la versión base de **PEAR**. Esto nos permitirá instalar nuevos paquetes y actualizar los ya existentes. Normalmente, encontraremos el directorio PEAR dentro de la carpeta **PHP** (aunque esto varía de acuerdo con cada instalación y es un aspecto configurable por el usuario).

The screenshot shows the official PEAR website homepage. At the top, there's a navigation bar with links for Main, Support, Documentation, Packages, Package Proposals, Developers, and Bugs. Below the navigation is a search bar. The main content area has a section titled "PEAR - PHP Extension and Application Repository". It includes a "What is it?" section with a brief description of PEAR as a framework for reusable PHP components, and links to "Installing PEAR" and "installing pear packages". There's also a "Hot off the Press" section with a note about RSS feeds and a command-line example for fetching them. On the right side, there are three columns: "Recent Releases" listing several packages like HTML_QuickForm, MDB2_TableBrowser, and Net_DNSBL; "Popular Packages" listing Net_Socket, Mail, DB_DataObject, Crypt_CHAP, and HTTP_Download; and "Recently Proposed" listing various proposed packages. At the bottom left, there's a "PEAR Community" sidebar with links to mailing lists, IRC, LinkedIn, Ohloh, Twitter, Identica, Facebook, and the wiki, along with a "Commit Activity Timeline" chart.

Figura 1. Desde el sitio oficial de PEAR, podemos obtener información acerca de las mejoras introducidas en las últimas versiones lanzadas.

Para comenzar a instalar nuevas opciones, debemos contar con el administrador de paquetes de PEAR, incluido desde las últimas versiones de PHP en la distribución oficial. Si no es nuestro caso (PHP versión inferior a 4.3.0) o simplemente queremos actualizar el administrador para contar con la última versión, debemos ejecutar a través del intérprete PHP una página a la que denominaremos **go-pear.php**. Podemos obtener su contenido desde <http://pear.php.net/go-pear>.

```
C:\> php go-pear.php
```

Durante el proceso de instalación, el intérprete nos pedirá que definamos algunas opciones generalmente relativas a las rutas de los directorios de PHP y de PEAR. Luego de completado el proceso, en forma opcional, podemos incluir el directorio de PEAR en la variable de entorno (**PATH**, esto es sólo para sistemas Windows).

Lo hacemos desde **Inicio/Panel de Control/Sistema/Opciones Avanzadas/Variables de Entorno** o, simplemente, ejecutando el archivo **PEAR_ENV.reg**, ubicado dentro del directorio de instalación de PHP.

```
<?php //: echo; echo "YOU NEED TO RUN THIS SCRIPT WITH PHP!"; echo; echo "Point your webbrowser to it or run: php -q go-pear.php"; echo; exit # -*- PHP -*-  
#  
# The PEAR installation wizard, both webbased or command line.  
#  
# Webbased installation:  
# 1) Download this file and save it as go-pear.php  
# 2) Put go-pear.php on your webserver, where you would put your website  
# 3) Open http://yourdomain.example.org/go-pear.php in your browser  
# 4) Follow the instructions, done!  
#  
# Command-line installation (for advanced users):  
# 1) Download this file and save it as go-pear.php  
# 2) Open a terminal/command prompt and type: php -q go-pear.php  
# 3) Follow the instructions, done!  
#  
# Notes:  
# * Get the latest go-pear version from http://pear.php.net/go-pear  
# * This installer requires PHP 4.3.0 or newer.  
# * On windows, the PHP CLI binary is php.exe, don't forget the -q option if using the CGI binary.  
# * The default for the command-line installation is a system-wide configuration file. For a local install use: php -q go-pear.php local  
  
/*  
 * go-pear is the online PEAR installer: just download it and run it  
 * (through a browser or command line), it will set up a minimal PEAR  
 * installation that will be ready for immediate use.  
 *  
 * @license http://www.php.net/license/2_02.txt PHP License 2.02  
 * @version CVS: $Id: go-pear 281637 2009-06-04 08:51:45Z clockwerk $  
 * @link http://pear.php.net/package/pearweb_gopear  
 * @author Tomas V.V.Cox <tova@ideonet.com>  
 * @author Stig Bakken <ssb@php.net>  
 * @author Christian Dickmann <dickmann@php.net>  
 * @author Pierre-Alain Joye <pierre@php.net>  
 * @author Greg Beaver <cellog@php.net>  
 * @author Tias Guns <tias@ulyssis.org>  
 */  
  
$name = php_sapi_name();
```

Figura 2. Podemos obtener la última versión del código para generar el administrador de paquetes de PEAR desde el sitio <http://pear.php.net/go-pear>.

Para una instalación remota, podemos invocar el archivo **go-pear.php** a través del navegador y seguir las instrucciones:

```
http://www.nombre-sitio.com/go-pear.php
```

Una vez que concluimos la instalación del administrador, ya estamos en condiciones de utilizar el comando **pear** a través de una terminal. La sintaxis para instalar nuevos paquetes o actualizar los ya existentes es la siguiente:

```
C:\> pear install nombrePaquete
```

En lugar del nombre del paquete, podemos realizar instalaciones a partir de archivos, incluyendo la ruta correspondiente:

```
C:\> pear install nombreArchivoPaquete.tgz
```

Mientras la primera opción necesita descargar el paquete desde el sitio <http://pear.php.net>, la segunda puede ser utilizada offline. En el sitio oficial de PEAR, están disponibles los archivos correspondientes a cada uno de ellos. Otra posibilidad es a través de la opción **download** del comando **pear**:

```
C:\> pear download nombrePaquete
```

Puede que un paquete tenga diferentes tipos de versiones (**alpha**, **beta**, **devel**, **stable**, **snapshot**). El administrador de paquetes nos informará al respecto para que nos definamos por una u otra. Para obtener el listado de paquetes disponibles, podemos acceder a <http://pear.php.net/packages.php> o desde la línea de comandos:

```
C:\> pear remote-list
```

Otras opciones disponibles:

- Información acerca de un paquete:

```
C:\> pear info nombrePaquete
```

- Desinstalar paquetes:

```
C:\> pear uninstall nombrePaquete
```

- Paquetes instalados:

```
C:\> pear list
```

III PECL

Según en el lenguaje en el que estén desarrolladas, las clases o extensiones mantenidas por los responsables de PHP estarán contenidas en PEAR (escritas en lenguaje C/C++) o **PECL** (*PHP Extension Community Library*, escritas en lenguaje PHP). Podemos encontrar más información acerca de esta opción en el sitio web <http://pecl.php.net>.

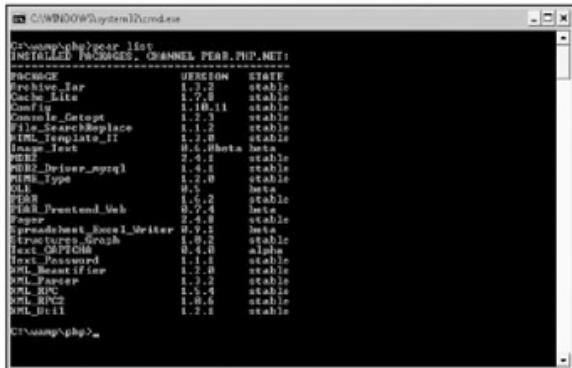


Figura 3. El comando pear puede invocarse desde la línea de comandos del sistema operativo y admite múltiples posibilidades para administrar las opciones instaladas.

- Paquetes que han sufrido actualizaciones:

```
C:\> pear list-upgrades
```

- Actualizar todas las opciones instaladas:

```
C:\> pear upgrade-all
```

- Sintaxis de las opciones:

```
C:\> pear help nombreComando
```

Existen múltiples opciones, además de las vistas, que podremos consultar en el manual oficial. PEAR está construido de manera tal que un paquete depende de otros para su funcionamiento. Este hecho nos será advertido por el administrador de paquetes al intentar, por ejemplo, instalar un paquete sin contar con las dependencias correspondientes. Para instalar todas las dependencias de un paquete de manera automática, contamos con la opción **—alldeps** (o su equivalente **—a**):

```
C:\> pear install --alldeps nombrePaquete
```

```
C:\> pear install -a nombrePaquete
```

Para instalaciones remotas, podemos visitar el apartado destinado al respecto en <http://pear.php.net/manual/es/installation.shared.php>.

OPCIONES DISPONIBLES

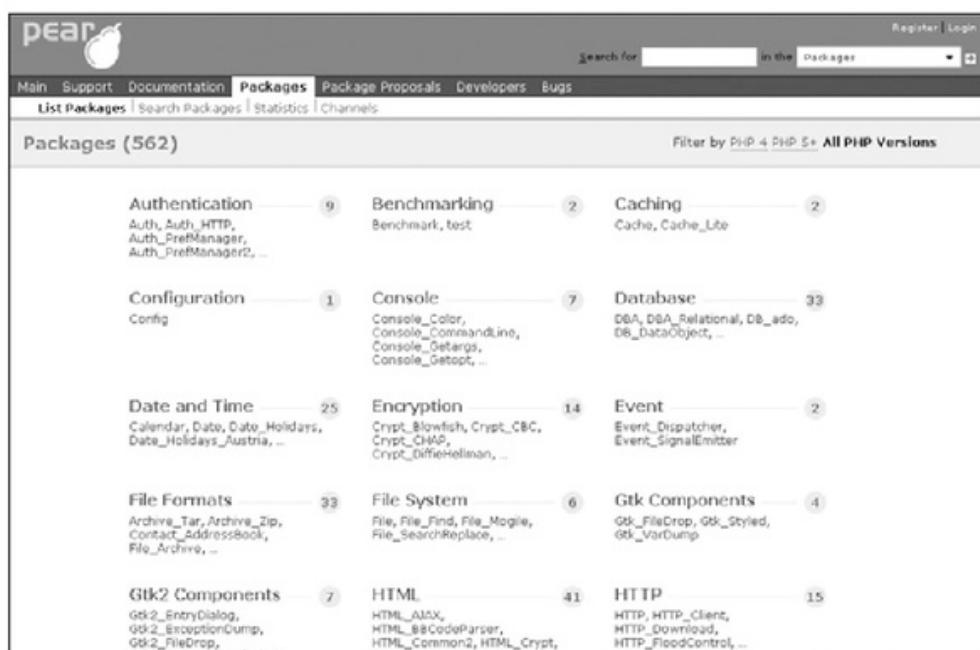
PEAR mantiene alrededor de 457 paquetes distribuidos en categorías. Cada una tiene un propósito particular, y un paquete puede depender de otro de una categoría distinta para funcionar. En la siguiente lista vemos un breve resumen de ellas:

NOMBRE	DESCRIPCIÓN
Authentication	Autenticación de accesos.
Benchmarking	Pruebas de comportamiento de aplicaciones, por ejemplo velocidad de respuesta ante peticiones.
Caching	Implementación y administración de caché.
Configuration	Manejo de archivos de configuración en formato XML o en estructuras PHP, por ejemplo.
Console	Aplicaciones PHP para línea de comandos.
Database	Capas de abstracción para acceso a datos.
Date and Time	Fecha y hora.
Encryption	Encriptación y desencriptación de datos.
Event	Comunicación entre objetos a partir de eventos.
File Formats	Trabajo con diversos formatos de archivos como hojas de estilo XLS, tags MP3, archivos CVS, etcétera.
File System	Funciones para trabajo con archivos (buscar, buscar y reemplazar, directorios, rutas, etcétera).
Gtk	Generación de aplicaciones PHP-Gtk.
Gtk2	Generación de aplicaciones PHP-Gtk2.
HTML	Generación de HTML (formularios, menús, tablas, templates, paginadores, etcétera).
HTTP	Trabajo con el protocolo HTTP (subida y bajada de datos, peticiones, sesiones, etcétera).
Images	Manipulación de imágenes (código de barras, gráficos, etiquetas, etcétera).
Internationalization	Internacionalización de aplicaciones.
Logging	Administración de ingresos en aplicaciones.
Mail	Funcionalidades relativas a la generación y envío de correos electrónicos.
Math	Paquetes relacionados con funciones matemáticas.
Networking	Trabajos con diversos protocolos de red.
Numbers	Funciones numéricas.
Payment	Comercio electrónico.
PEAR	Administración de paquetes PEAR.
PHP	Documentación de sistemas, embellecedores de código, compatibilidad entre versiones, etcétera.
QA Tools	Definición de la calidad de una aplicación.
Science	Propósitos científicos.
Streams	Envío y recepción de datos.
Structures	Tratamiento de estructuras tabulares y de árbol en diversos formatos.
System	Utilidades de sistema (unidades, directorios, etcétera).

NOMBRE	DESCRIPCIÓN
Text	CAPTCHA, arte ASCII, generación de contraseñas, etcétera.
Tools and Utilities	SVN, MIME, etcétera.
Validate	Validaciones específicas y orientadas según regiones.
Web Services	Servicios web (YouTube, Yahoo, pronóstico del clima, validador W3C, Delicious, Ebay, Technorati, etcétera).
XML	Parseo y conversión desde/hacia XML.

Tabla 1. Categorías de paquetes disponibles en PEAR.

Podemos descubrir las innumerables alternativas ofrecidas y puestas a disposición del desarrollador en el sitio oficial <http://pear.php.net>.

**Figura 4.** PEAR divide los paquetes disponibles en categorías, permitiendo un acceso rápido e intuitivo a ellos.

Una de sus características es la escalabilidad: el método de instalación permite que las nuevas opciones no tengan conflictos con las que ya están. La idea es descargar sólo aquellos paquetes que se van a utilizar.

III CRIPTOGRAFÍA EN PEAR

Es importante destacar que PEAR nos provee de paquetes relacionados con la criptografía en aplicaciones web. Podemos obtener más información acerca de las opciones disponibles en esta librería si visitamos el sitio que se encuentra en http://pear.php.net/package/Crypt_HMAC y también en http://pear.php.net/package/Crypt_RSA.

UTILIZACIÓN

Así como en sistemas Windows contamos con las variables de entorno para disponer de programas accediendo por la línea de comandos desde cualquier directorio, la directiva **include_path** nos permite realizar una tarea similar para el uso de los paquetes instalados en nuestros scripts. Al momento de la instalación, esto se realiza de manera automática, por lo que deberíamos contar en el archivo **php.ini** con una línea como la siguiente (debemos tener presente que, para ver reflejados los cambios realizados en el archivo **php.ini**, tendremos que reiniciar el servidor web y actualizar la página actual):

```
include_path = ".;c:\php\includes;c:\php\PEAR"
```

Las rutas dependerán de cada sistema en particular, pero la idea es incluir el directorio base de PEAR. Si navegamos por el sistema de archivos, más específicamente por el directorio de instalación de PHP, podremos llegar a visualizar la carpeta en cuestión, que generalmente se denomina **Pear**. La ruta por incluir en el **php.ini** deberá ser absoluta.



Figura 5. Además de las múltiples directivas de configuración, el archivo **php.ini** mantiene una sección dedicada a definir los directorios de inclusión de archivos.

III CARACTÉRISTICA DE PEAR

Es relevante saber que PEAR es un proyecto supervisado por los desarrolladores del lenguaje PHP, algo que transmite seguridad a aquellos que piensen incorporar las clases disponibles en sus proyectos. Además, la estructura de PEAR permite a los propios usuarios aplicar mejoras sobre cada opción y hacerse cargo del mantenimiento.

ERRORES

Más allá de las opciones que instalamos de forma manual y personalizada, PEAR dispone de algunas clases base de propósito general, por ejemplo **PEAR_Error**. En el caso de producirse un error, las clases PEAR devuelven un objeto de este tipo, el cual puede reconocerse mediante el método **isError** que recibe como argumento la variable por verificar. El método **getMessage** nos permite obtener el mensaje descriptivo acerca del error cometido.

```
if(PEAR::isError($objeto)) {
    die($objeto->getMessage());
}
```

IMPLEMENTACIONES

Como vimos, PEAR ofrece una o más respuestas a cada demanda que pueda llegar a surgir en un desarrollo. Para observar el modo de funcionamiento de algunas de las clases disponibles, tomaremos como ejemplo tres paquetes muy populares: **MDB2** (una poderosa capa de abstracción para acceso a datos), **Spreadsheet_Excel_Writer** (para generar documentos de extensión **xls** desde scripts PHP) y **HTML_QuickForm** (para facilitar la creación y validación de formularios HTML).

Acceso a datos con MDB2

PHP pone a nuestra disposición un número importante de extensiones para acceder a distintos servidores de bases de datos. Cada una mantiene sus propios objetos, métodos, propiedades, etcétera, algo que imposibilita cambiar fácilmente de motor sin tener que reescribir la aplicación, actualizando los nombres de las funciones, los argumentos a cada una, etcétera. Por no hablar de las diferencias para



VELOCIDAD

Debemos tener en cuenta que si bien podríamos suponer que utilizar una capa de abstracción para acceder a una base de datos ralentiza el desempeño de una aplicación, esto no es tan así. En forma evidente, la velocidad de acceso disminuye un poco, pero es cierto que para la mayoría de los desarrollos, el cambio es casi imperceptible.

con el lenguaje SQL. Para poder acceder a distintas bases de datos utilizando un mismo código, existen las llamadas **capas de abstracción para acceso a datos**, que proveen clases universales para realizar consultas, ejecutar instrucciones, recuperar resultados, recorrerlos, etcétera, permitiendo lograr un nivel de portabilidad destacado en las aplicaciones.

Main Support Documentation Packages Package Proposals Developers Bugs

List Packages | Search Packages | Statistics | Channels

Top Level :: Database

Package Information: MDB2

Main Download Documentation Bugs Trackbacks

>> Summary

database abstraction layer

>> Current Release

2.5.0b3 (beta) was released on 2010-08-29 ([Changelog](#))
2.4.1 (stable) was released on 2007-05-03 ([Changelog](#))

>> License

BSD License

>> Bug Summary

- Package Maintenance Rank: 71 of 202 packages with open bugs
- Number of open bugs: 14 (296 total bugs)
- Average age of open bugs: 313 days
- Oldest open bug: 832 days
- Number of open feature requests: 12 (92 total feature requests)

[Report a new bug to MDB2](#)

>> Description

PEAR MDB2 is a merge of the PEAR DB and Metabase php database abstraction layers. It provides a common API for all supported RDBMS. The main difference to most other DB abstraction packages is that MDB2 goes much further to ensure portability. MDB2 provides most of its many features optionally that can be used to construct portable SQL statements:

- * Object-Oriented API
- * A DSN (data source name) or array format for specifying database servers
- * Datatype abstraction and on demand datatype conversion
- * Various optional fetch modes to fix portability issues
- * Portable error codes
- * Sequential and non sequential row fetching as well as bulk fetching
- * Ability to make buffered and unbuffered queries

Figura 6. El paquete MDB2 antecede a otro muy popular llamado PEAR::DB.

Las capas de abstracción para acceso a datos permiten conectarse, a partir de un mismo código, con múltiples sistemas gestores de bases de datos sin cambiar una línea. Esto deriva en la no dependencia de una herramienta en particular, facilitando enormemente la migración de un motor a otro. La opción **MDB2** de PEAR junto con **ADObd**, **PHPLib** y **Metabase** son quizás las más populares alternativas. MDB2 soporta, hasta el momento, las siguientes bases:

III OPCIONES

Es importante señalar que uno de los casos de uso que podría vincularse a la utilización de las capas de abstracción para acceso a datos es la necesidad de programar un sistema que funcione por medio de un gestor de bases de datos, que no se encuentre disponible en la máquina local que es utilizada durante la etapa de desarrollo.

- MySQL
- PostgreSQL
- Oracle
- Frontbase
- Querysim
- Interbase/Firebird (PHP versión 5)
- MSSQL
- SQLite

ADODB Database Abstraction Library for PHP (and Python)

© 2000-2009 John Lim (jlim@jlimsoft.com). All rights reserved.

[Download](#) [Documentation](#) [Forums \(post bug reports here\)](#) [Changelog](#) [FAQ](#)

ADODB is a database abstraction library for PHP. There is also a Python version; see the [ADODB for Python docs](#).

The PHP version currently supports an amazing number of databases, thanks to the wonderful ADODB community: MySQL, PostgreSQL, Interbase, Firebird, Informix, Oracle, MS SQL, Foxpro, Access, ADO, Sybase, FrontBase, DB2, SAP DB, SQLite, Netezza, LDAP, and generic ODBC, ODBTP. The Sybase, Informix, FrontBase and PostgreSQL, Netezza, LDAP, ODBTP drivers are community contributions. Here is the [complete list of drivers](#).

Many popular web applications such as [ACID](#), [Zikula](#), [Postluke](#), [Xeraya](#), [phoWiki](#), [Mambo](#), [PHP-GACL](#), [TikiWiki](#), [eGroupWare](#) and [phpList](#), [App Server](#) are using ADODB as their database abstraction layer. Some reasons why ADODB is popular include:

- Designed for **speed**. It is probably the fastest open source database abstraction library available for PHP. See [benchmarks](#).
- Provides **extensive portability support** such as date and type-handling and portable schema creation. See [portable sql tips](#).
- Supports many **enterprise features** such as database backed sessions (with session expiry notification), SQL code generation, pivot tables, SELECT LIMIT emulation for all databases, performance monitoring.
- **Easy to learn**, especially if you have Windows' programming experience, as it uses many ADO conventions.
- **Extensive QA**, every release is unit-tested on Access, MySQL, PostgreSQL, MS SQL, Oracle 11g.
- **Mature**, continuously developed since August 2000. Has a **large community** of users.
- Powerful **Active Record** support. See [docs](#).
- Very **reasonable licensing terms** (BSD). This means that you can incorporate (and even compile) it into your software applications **royalty-free** without asking the author's permission, provided you include license.txt in your release. Also dual-licensed (Lesser GPL).

PHP Code Samples

```
include('path/to/adodb/ini.php');
$DB = NewADOConnection('mysql');
$DB->Connect('Server', 'User', 'Pwd', $DB);
```

Figura 7. ADODB es una popular alternativa a MDB2. Cuenta con una gran comunidad de usuarios y presenta una sintaxis similar a la del lenguaje ASP.

Si bien no es la única alternativa, desde PEAR se recomienda a MDB2 como la más viable para interactuar con bases de datos. Para instalar el paquete, debemos ejecutar la siguiente línea desde una consola:

```
c:\> pear install MDB2
```

III CONSULTAS PREPARADAS

MySQL que podemos encontrar en la dirección web www.mysql.com, es un popular motor de bases de datos que, sin embargo, brinda soporte para implementar consultas preparadas desde hace mucho tiempo. Se implementó desde la versión 4.1.2, y, en el manual oficial de PHP, se incluyen ejemplos de uso utilizando la extensión mysql.

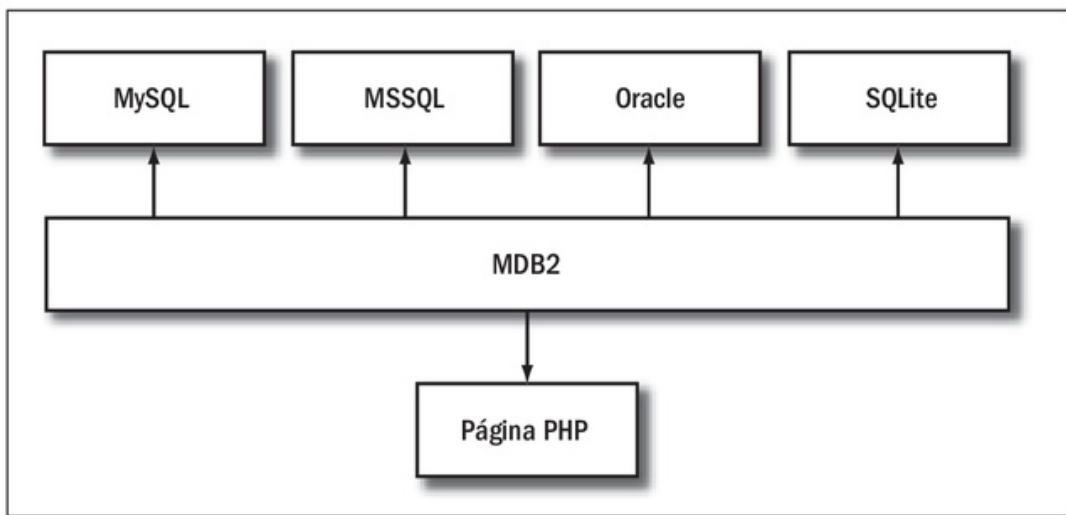


Figura 8. MDB2 permite acceder a distintas bases de datos emulando las características no disponibles de cada una.

Al hacer esto, sólo instalamos las clases básicas de MDB2, para poder acceder a una base de datos a través de este paquete, agregando los drivers por separado:

```
c:\> pear install MDB2#mysql
```

```
c:\> pear install MDB2#pgsql
```

```
c:\> pear install MDB2#sqlite
```

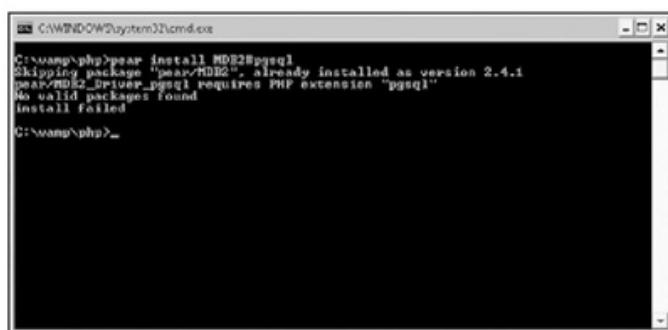


Figura 9. Para instalar un driver específico, la extensión PHP asociada debe estar habilitada.

Para establecer una conexión, necesitaremos crear un **DSN** (*Data Source Name*), que es una cadena de caracteres compuesta por varias partes:

```
gestor://usuario:contraseña@servidor/base
```

En donde, **gestor** puede tomar uno de los siguientes valores:

CONSTANTE	BASE DE DATOS
fbsql	FrontBase
ibase	InterBase / Firebird
mssql	Microsoft SQL Server
mysql	MySQL
mysqli	MySQL (para versiones 4.1 y superiores)
oci8	Oracle 7/8/9/10
pgsql	PostgreSQL
querysim	QuerySim
sqlite	SQLite 2

Tabla 2. Constantes de las bases de datos soportadas.

Algunos ejemplos de conexión por DSNs usando distintos tipos de bases de datos:

mssql://user170:cV6yta1@localhost/clientes

pgsql://usuario:pass@127.0.0.1/base

sqlite:///ruta/al/archivo.db?mode=0644

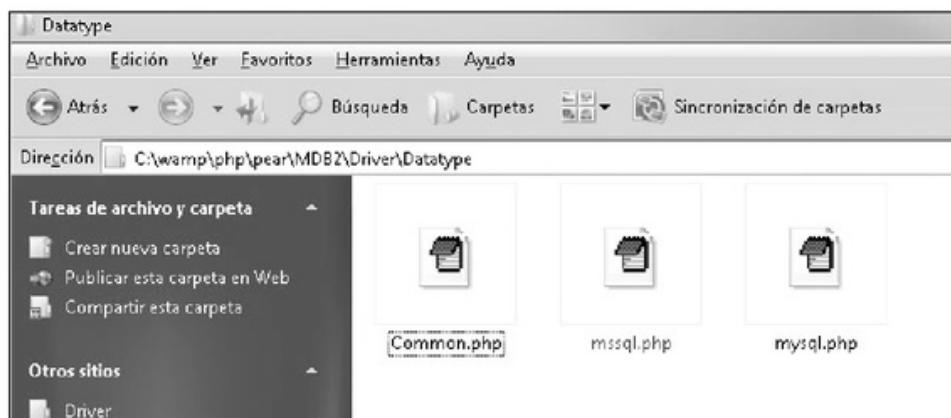


Figura 10. Es posible instalar drivers de distintas bases de datos y, una manera de conocer los disponibles, es inspeccionando el directorio driver/datatype.

Una segunda sintaxis está dada por la inclusión del protocolo utilizado:

gestor://usuario:contraseña@protocol(opciones)/base

Por ejemplo, si utilizamos **sockets** para establecer la conexión:

```
mysql://root@pass(/ruta/al/socket)/pear
```

Una vez creado el DSN, nos servirá para pasarlo como argumento a alguno de los métodos disponibles para establecer la conexión: **factory**, **connect** o **singleton**:

```
<?php

$driver = "mysql";
$hostname = "localhost";
$database = "mdb2";
$username = "root";
$password = "";

$dsn = "$driver://{$username}{$password}@{$hostname}/{$database}";

$con = MDB2::factory($dsn);

if(PEAR::isError($con)) {
    die($con->getMessage());
}

$con->disconnect();

?>
```

En lugar de una cadena de caracteres, estos métodos pueden recibir como argumento un array cuyas claves serán las partes de un DSN:

```
$dsn = array(
    'phptype' => false, //gestor
    'username' => false, //nombre de usuario
    'password' => false, //contraseña
    'protocol' => false, //protocolo
    'hostspec' => false, //servidor
    'port'      => false, //puerto
    'socket'    => false, //socket
```

```

'database' => false //nombre de la base
);

```

En cualquiera de las dos alternativas, si intentamos conectar a una base SQLite, debemos incluir la opción **mode** (que indica el modo de acceso al archivo que contiene la base), como vemos en el código siguiente:

```

$dsn = array(
    'phptype' => 'sqlite',
    'mode'      => '0666',
    'database'  => 'clientes'
);

```

Figura 11. SQLite (www.sqlite.org) es una de las bases de datos disponibles para acceder desde MDB2. Es orientada a archivos, y su popularidad crece día a día.

Podemos obtener una lista completa de las opciones y las sintaxis disponibles para construir un DSN en el manual oficial de PEAR.

Si en alguno de los argumentos contenidos en un DSN existen caracteres delimitadores, deberemos reemplazarlos según corresponda:

CARÁCTER	REEMPLAZO
:	%3a
+	%2b
?	%3f
/	%2f
(%28
=	%3d
@	%40
)	%29
&	%26

Tabla 3. Caracteres delimitadores en MDB2.

Existen dos métodos para ejecutar instrucciones SQL: **query** (para aquellas que podrían llegar a devolver filas, por ejemplo, **select**) y **exec** (para las que no, por ejemplo, **insert**, **delete**, **update**, **create**, etcétera).

El método **query** recibe como argumento la instrucción y devuelve un objeto **MDB2_Result**, si todo funcionó bien, o un objeto **MDB2_Error**, si ocurrió un error.

```
create database ventas;

use ventas;

create table vendedores (
    idVendedor int(11) primary key not null auto_increment,
    apellidoVendedor varchar(255),
    nombreVendedor varchar(255)
);
```

```
<?php

require_once 'MDB2.php';
```

III ADODB

ADODb que encontramos en la dirección web <http://sourceforge.net/projects/adodb> es una opción alternativa para que podamos acceder a distintos motores de bases de datos utilizando el mismo código que hemos desarrollado. Su sintaxis es similar a la utilizada por el lenguaje ASP versión 3.0 y posibilita acceder a una gran cantidad de bases de datos.

```
$mdb2 = MDB2::connect('mysql://root@localhost/ventas');

if (PEAR::isError($mdb2)) {
    die($mdb2->getMessage());
}

$res = $mdb2->query('SELECT * FROM vendedores');

if (PEAR::isError($res)) {
    die($res->getMessage());
}

echo 'Se ejecuto correctamente la instruccion';

?>
```

El método **exec** recibe como argumento la instrucción y devuelve un objeto **MDB2_Result**, si todo funcionó bien, o un entero que representa el número de filas afectadas.

```
<?php

require_once 'MDB2.php';

$ mdb2 = MDB2::connect('mysql://root@localhost/ventas');

if (PEAR::isError($mdb2)) {
    die($mdb2->getMessage());
}

$res = $mdb2->exec("INSERT INTO vendedores (apellidoVendedor,
    nombreVendedor) VALUES ('A', 'B')");

if (PEAR::isError($res)) {
    die($res->getMessage());
}

echo "$res fila/s afectada/s";

?>
```

Como observamos en el capítulo anterior, el filtrado de variables que forman parte de una instrucción SQL es de suma importancia para evitar posibles ataques (**SQL Injection**).

Al respecto, **MDB2** provee el método **quote** que recibe cuatro argumentos para su funcionamiento:

- El valor por escapar (obligatorio).
- El tipo de dato correspondiente (opcional).
- Si encerrar entre comillas los valores o no (opcional).
- Si escapar los caracteres _ y % o no (opcional).

```
$mdb2->quote($valor, 'integer', false, false);
```

```
$mdb2->quote($valor, 'text', true, false);
```

```
$mdb2->quote(date("Y-m-d"), 'date', true, false);
```

Algunos de los tipos de datos soportados por **MDB2** son:

- **integer**
- **float**
- **decimal**
- **text**
- **timestamp**
- **date**
- **time**
- **boolean**
- **blob**
- **clob**

De manera alternativa, podemos utilizar el método **escape** para escapar valores directamente sin agregar comillas. Para manejar los resultados obtenidos a partir de un objeto **MDB2_Result** existen cuatro métodos disponibles:

- **fetchOne**: devuelve en cada llamada el primer campo de la fila de resultados:

```
$res = $mdb2->query("SELECT apellidoVendedor FROM vendedores");
```

```

if (PEAR::isError($res)) {
    die($res->getMessage());
}

while (($apellidoVendedor = $res->fetchOne())) {
    echo "<li>".$apellidoVendedor;
}

```

- **fetchRow**: retorna la fila completa de resultados:

```

$res = $mdb2->query("SELECT nombreVendedor, apellidoVendedor FROM
                      vendedores");

if (PEAR::isError($res)) {
    die($res->getMessage());
}

while (($row = $res->fetchRow())) {
    echo "<li>".$row[0].", ".$row[1];
}

```

- **fetchCol**: regresa un array que contiene valores de la primera columna del resultado:

```

$res = $mdb2->query("SELECT nombreVendedor, apellidoVendedor FROM
                      vendedores");

if (PEAR::isError($res)) {
    die($res->getMessage());
}

```



LIBERAR MEMORIA

Debemos tener en cuenta que una vez recuperados los resultados obtenidos a partir de una consulta, puede ser de mucha utilidad liberar la memoria utilizada en el proceso, de esta forma podremos agilizar la ejecución de la página. Podemos lograrlo mediante el método **free** (que no recibe ningún argumento) del objeto **MDB2_Result**.

```
echo '<pre>';
print_r($res->fetchCol());
echo '</pre>';
```

- **fetchAll:** reintegra un array con **todas** las filas del resultado:

```
$res = $mdb2->query("SELECT * FROM vendedores");

if (PEAR::isError($res)) {
    die($res->getMessage());
}

echo '<pre>';

print_r($res->fetchAll());
echo '</pre>';
```

Podemos definir el tipo de acceso a los campos contenidos en una fila de resultados a partir de constantes:

CONSTANTE	DESCRIPCIÓN
MDB2_FETCHMODE_ORDERED	Valor por defecto, índices numéricos.
MDB2_FETCHMODE_ASSOC	Índices asociativos, nombres de columnas.
MDB2_FETCHMODE_OBJECT	Objetos, cada columna es una propiedad de la fila devuelta.

Tabla 4. Constantes para acceder a valores de filas.

```
$res = $mdb2->query("SELECT nombreVendedor, apellidoVendedor FROM
vendedores");

if (PEAR::isError($res)) {
```



TRANSACCIONES

Una de las funcionalidades brindadas por MDB2 es la implementación de transacciones mediante los métodos denominados **beginTransaction**, **rollback** y **commit**. Debemos saber que la utilización de las transacciones nos permite lograr la ejecución de todas las instrucciones, o ninguna en caso de error, para asegurar la integridad.

```

        die($res->getMessage());
    }

while (($row = $res->fetchRow(MDB2_FETCHMODE_ASSOC))) {
    echo "<li>".$row[nombrevendedor];
}

```

Podemos definir un método por defecto a través de **setFetchMode**:

```

$ mdb2->setFetchMode(MDB2_FETCHMODE_ASSOC);

$res = $mdb2->query("SELECT nombreVendedor, apellidoVendedor FROM
vendedores");

if (PEAR::isError($res)) {
    die($res->getMessage());
}

while (($row = $res->fetchRow())) {
    echo "<li>".$row[nombrevendedor];
}

```

En MySQL, contamos con la cláusula **LIMIT** para recuperar una página de resultados determinada por una fila de inicio y una cantidad de registros a partir de ella, aquí vemos un ejemplo con 5 y 10 registros respectivamente:

```
SELECT * FROM nombreTabla WHERE campo1 > campo2 LIMIT 5, 10;
```

Esta opción es específica de este motor, pero aun así MDB2 nos da la opción de utilizarla a través del método **setLimit**:

```

$ mdb2->setLimit(10, 5);
$res = $mdb2->query("SELECT * FROM nombreTabla WHERE campo1 > campo2");

```

La llamada a este método sólo tiene validez para la siguiente consulta que se realice. En el mismo sentido, contamos con **limitQuery** (disponible si cargamos el módulo **Extended**), que tiene una sintaxis diferente:

```
$mdb2->loadModule('Extended');
$res = $mdb2->limitQuery("SELECT * FROM nombreTabla WHERE campo1 > campo2",
    null, 10, 5);
```

Los objetos **MDB2_Result** poseen otros métodos útiles para conocer las características del conjunto de resultados devueltos:

MÉTODO	DESCRIPCIÓN
numRows	Número de filas devueltas.
numCols	Número de columnas devueltas.
rowCount	Fila actual.
getColumnNames	Nombres de las columnas devueltas.

Tabla 5. Algunos de los métodos disponibles para **MDB2_Result**.

Existen métodos que nos permiten realizar consultas y recibir el resultado en una misma acción. Reciben como argumento la instrucción SQL por ejecutar:

MÉTODO	DESCRIPCIÓN
queryOne	Similar a query más fetchOne .
queryRow	Similar a query más fetchRow .
queryCol	Similar a query más fetchCol .
queryAll	Similar a query más fetchAll .

Tabla 6. Métodos disponibles para recuperar registros incluidos en un resultado.

Al momento de ejecutar una instrucción SQL, se completan una serie de pasos que conforman la comunicación entre una aplicación y una base de datos:

- Se crea la instrucción en PHP.
- Se la envía al servidor de bases de datos.
- Éste controla que la instrucción sea válida.
- Devuelve un resultado.
- Desde el script, se procesan los datos obtenidos.

Por todo esto, podríamos decir que ante cada instrucción SQL se genera una petición al servidor, algo que haría volver más pesado el desempeño de la aplicación en caso de tener que ejecutar un gran número de sentencias. Las versiones más actuales de los servidores de bases de datos permiten procesar las llamadas **consultas preparadas** (*prepared statements*), cuyo funcionamiento en el ámbito de una aplicación es el siguiente:

- El script PHP envía un molde de la instrucción SQL, sin valores reales.
- El servidor de bases de datos lo almacena, previa validación.
- Desde el script se envían sólo las partes variables de la instrucción.

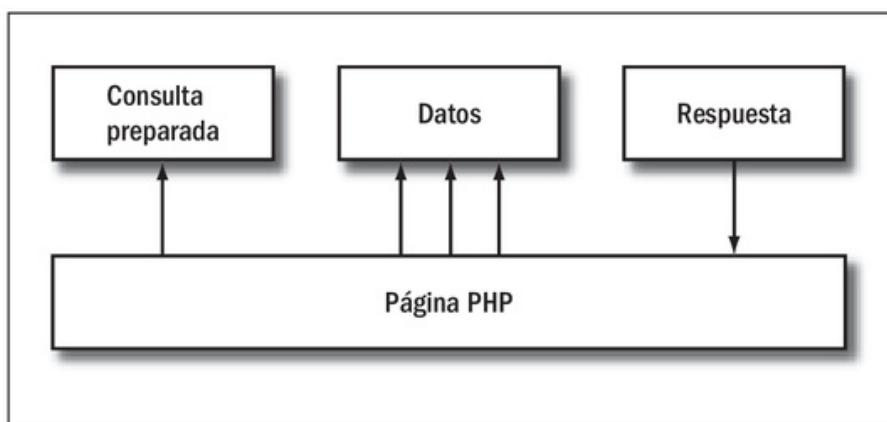


Figura 12. El uso de consultas preparadas es una característica importante dentro de las bases de datos profesionales.

MDB2 pone a nuestra disposición dos métodos para implementar consultas preparadas: **prepare** (para preparar la instrucción y obtener un objeto de tipo **MDB2_Statement_Common**) y **execute** (para ejecutar la consulta preparada).

```

$tipos = array('text', 'text');

$cp = $mdb2->prepare('INSERT INTO vendedores (apellidoVendedor,
    nombreVendedor) VALUES (?, ?)', $tipos, MDB2_PREPARE_MANIP);

$cp->execute(array('x', 'x'));
$cp->execute(array('y', 'y'));

$cp->execute(array('z', 'z'));

```

El método **prepare** recibe como argumentos la instrucción, un array que contiene los tipos de datos de cada campo por reemplazar en el SQL (o **true** para autodetectar), y una constante: para las instrucciones que devuelvan resultados, debemos utilizar **MDB2_PREPARE_RESULT**, mientras que para las demás **MDB2_PREPARE_MANIP**.

El método **execute**, por su parte, recibe un array con los datos por ingresar en la instrucción SQL. Devuelve como resultado el número de filas afectadas.

En el caso de las instrucciones que devuelven resultados, **prepare** puede recibir como tercer argumento, en lugar de **MDB2_PREPARE_MANIP**, un array con los tipos de datos devueltos (uno por columna) o **true** para autodetectar:

```
$cp = $mdb2->prepare('SELECT apellidoVendedor, nombreVendedor FROM vendedores WHERE idVendedor = ?', array('integer'), array('text', 'text'));

$res = $cp->execute(3);
$res = $cp->execute(2);
$res = $cp->execute(4);
```

Para tratar los resultados obtenidos, contamos con métodos como **getOne**, **getRow**, **getCol**, **getAll** y **getAssoc**, que nos permiten, a través del módulo **Extended**, recuperar filas y columnas:

```
$mdb2->loadModule('Extended');

$res = $mdb2->extended->getRow('SELECT * FROM vendedores WHERE idVendedor = ?', null, array(1), array('integer'));

echo '<pre>';
print_r($res);
echo '</pre>';
```

Podemos ejecutar más de una instrucción SQL al mismo tiempo a través del método **executeMultiple**, disponible si se habilita el módulo **Extended** de MDB2:

```
$mdb2->loadModule('Extended', null, false);

$datos[] = array('x', 'x');
$datos[] = array('y', 'y');
$datos[] = array('z', 'z');
```

III PAGINADOR

El paquete MDB2 no incluye funciones para paginar resultados, pero tenemos la posibilidad de acudir al paquete **Pager** para ordenar los resultados devueltos por una instrucción SQL y mostrarlos en páginas. Esta opción no se limita a trabajar con bases de datos, sino que se aplica a cualquier estructura de datos de PHP, por ejemplo arrays.

```
$cp = $mdb2->prepare('INSERT INTO vendedores VALUES (?, ?)');
$mdb2->extended->executeMultiple($cp, $datos);
```

Estas clases de instrucciones se ejecutan como transacciones, es decir que, si alguna falla, las demás se anulan.

Para las consultas preparadas, no es necesario aplicar el método **quote**.

Mediante los métodos **autoPrepare** y **autoExecute**, es posible ejecutar instrucciones **INSERT**, **UPDATE**, **DELETE** y **SELECT** sin escribirlas de manera completa, como tradicionalmente hacemos. Nuevamente, deberemos incluir el módulo **Extended**:

- **INSERT**

```
$mdb2->loadModule('Extended');

$cambios = array('idVendedor', 'apellidoVendedor', 'nombreVendedor');
$tipos = array('integer', 'text', 'text');
$valores = array(10, 'x', 'x');

$cp = $mdb2->extended->autoPrepare('vendedores', $cambios,
    MDB2_AUTOQUERY_INSERT, null, $tipos);
$res = $cp->execute($valores);
```

- **UPDATE**

```
$mdb2->loadModule('Extended');

$cambios = array('apellidoVendedor', 'nombreVendedor');
$tipos = array('text', 'text');
$valores = array('y', 'y');

$cp = $mdb2->extended->autoPrepare('vendedores', $cambios,
    MDB2_AUTOQUERY_UPDATE, 'idVendedor = 10', $tipos);
$res = $cp->execute($valores);
```

- **DELETE**

```
$mdb2->loadModule('Extended');
```

```
$cp = $mdb2->extended->autoPrepare('vendedores', null,
    MDB2_AUTOQUERY_DELETE, 'idVendedor = 10');
$res = $cp->execute();
```

El método **autoExecute**, por su parte, simula una llamada a **autoPrepare** y **execute** de una sola vez, pasando como argumento un array asociativo que contenga nombres de campos y valores:

```
$mdb2->loadModule('Extended');

$cambios = array('idVendedor' => 11, 'apellidoVendedor' => 'z',
    'nombreVendedor' => 'z');
$tipos = array('integer', 'text', 'text');

$mdb2->extended->autoExecute('vendedores', $cambios, MDB2_AUTOQUERY_INSERT,
    null, $tipos);
```

MDB2 mantiene muchas otras opciones que podremos consultar en el manual oficial de la aplicación, en donde además se incluyen ejemplos de uso y una definición detallada de la sintaxis de cada uno de los métodos disponibles.

Hojas de cálculo con **Spreadsheet_Excel_Writer**

PEAR incluye una opción llamada **Spreadsheet_Excel_Writer**, que nos servirá para exportar datos hacia archivos XLS a través del lenguaje PHP. La mayoría de las aplicaciones necesitan generar reportes en algún formato a partir de información almacenada en una base de datos. Los lenguajes que se ejecutan tomando como marco un navegador web, usualmente, no brindan soluciones específicas como las aplicaciones de escritorio, por lo que la alternativa más viable consiste en generar documentos que puedan ser visualizados o impresos desde herramientas externas al ámbito del navegador.

CREACIÓN DE TABLAS

Es importante señalar que si bien las bases de datos suelen crearse utilizando la sintaxis particular de un motor específico, MDB2 ofrece a través del método denominado **createTable** la posibilidad de definir tablas. Se encarga de recibir como argumentos el nombre de la tabla y un array multidimensional que contiene los campos.

Package Information: Spreadsheet_Excel_Writer

Summary
Package for generating Excel spreadsheets (.xls)

Current Release
0.9.2 (beta) was released on 2009-11-20 ([Changelog](#))

Bug Summary

- Package Maintenance Rank: 154 of 202 packages with open bugs
- Number of open bugs: 54 (267 total bugs)
- Average age of open bugs: 882 days
- Oldest open bug: 2253 days
- Number of open feature requests: 21 (34 total feature requests)

[Report a new bug to Spreadsheet_Excel_Writer](#)

Description
Spreadsheet_Excel_Writer was born as a porting of the Spreadsheet::WriteExcel Perl module to PHP. It allows writing of Excel spreadsheets without the need for COM objects. It supports formulas, images (BMP) and all kinds of formatting for text and cells. It currently supports BIFF5 format (Excel 5.0) - BIFF8 format with Unicode support is also available now but still in early stage.

Spreadsheet_Excel_Writer is outdated and needs a complete rewrite. If you want to help with this task please get in touch with us. Otherwise we don't recommend this package for new development.

Maintainers

- Franck Lefevre [WishList] (developer)
- Carsten Schmitz (lead)

More Information

- [Browse the source tree](#)
- [RSS release feed](#)
- [View the changelog](#)

Figura 13. La generación de documentos XLS permite a los programadores dar versatilidad a sus desarrollos.

A diferencia de otras opciones, **Spreadsheet_Excel_Writer** no requiere objetos **COM** para funcionar. Además, soporta fórmulas, imágenes, formato de celdas, múltiples hojas y todas las características admitidas hasta la versión 5.0 de Microsoft Excel (**BIFF5**). Podemos saber más acerca de las características del paquete en el sitio oficial de PEAR: http://pear.php.net/package/Spreadsheet_Excel_Writer.



Figura 14. Microsoft Excel es una de las alternativas más populares para manipular documentos XLS.

Esta opción depende de otro paquete llamado **OLE**, por lo que, para disponer de sus funcionalidades, debemos escribir desde una terminal:

```
C:\php> pear install OLE
C:\php> pear install Spreadsheet_Excel_Writer
```

El instalador nos informará acerca de las versiones más actuales para que las definamos, por ejemplo:

```
C:\php> pear install OLE-0.5
C:\php> pear install Spreadsheet_Excel_Writer-0.9.1
```

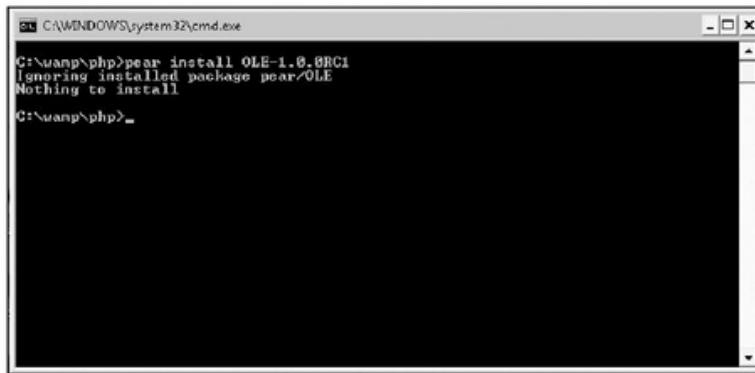


Figura 15. El administrador de paquetes de PEAR requiere que definamos la versión del paquete que queramos instalar.

Para comenzar a trabajar, debemos incluir la clase correspondiente e instanciar un objeto `Spreadsheet_Excel_Writer`:

```
require_once 'Spreadsheet/Excel/Writer.php';
$excel = new Spreadsheet_Excel_Writer();
```

Para añadir hojas (**sheets**) existe el método `addWorksheet`, que recibe como argumento opcional el nombre de la hoja:

```
$hoja =& $excel->addWorksheet();
```

El método `write` nos permite escribir valores en celdas y recibe tres argumentos: el número de fila (comenzando desde 0), el número de columna (comenzando desde 0) y el valor por escribir, como observamos en el siguiente ejemplo:

```
$hoja->write(0, 0, 'texto');
```

Una vez definido el documento, enviamos la salida al navegador:

```
$excel->send('nombreDocumento.xls');
```

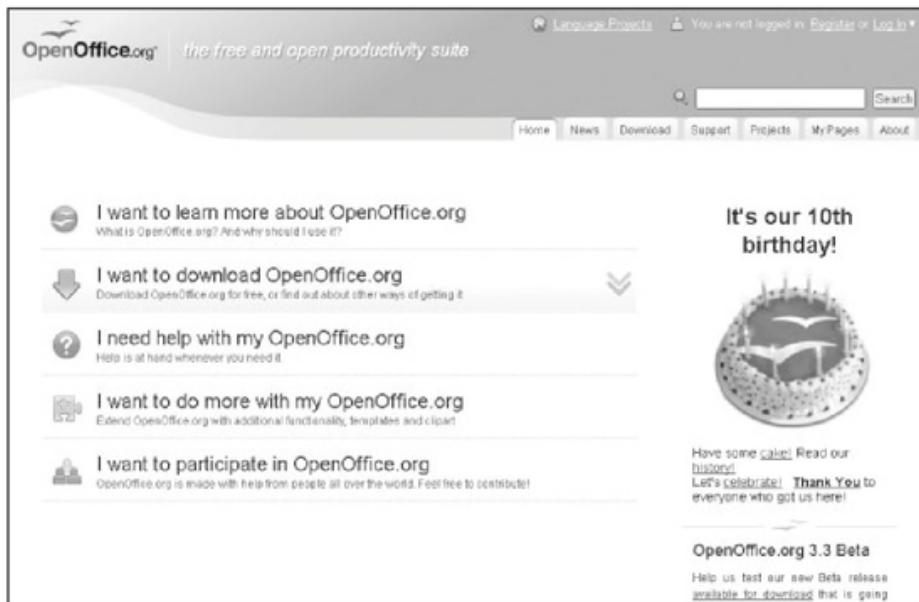


Figura 16. Microsoft Excel no es la única opción para trabajar con documentos XLS: otra muy interesante es **Calc** de OpenOffice.

Y destruimos el objeto:

```
$excel->close();
```

El método **write** recibe como cuarto argumento opcional una especificación del formato de la celda por modificar, que puede ser generado a partir del método **addFormat**, y aplicable sobre objetos de tipo **Spreadsheet_Excel_Writer**:

```
<?php

require_once 'Spreadsheet/Excel/Writer.php';
$excel = new Spreadsheet_Excel_Writer();
$excel =& $hoja->addWorksheet('nueva hoja');
$fondo =& $excel->addFormat();
$fondo->setBgColor('yellow');
```

```
$hoja->write(0, 0, 'texto', $fondo);

$excel->send('nombreDocumento.xls');
$excel->close();

?>
```

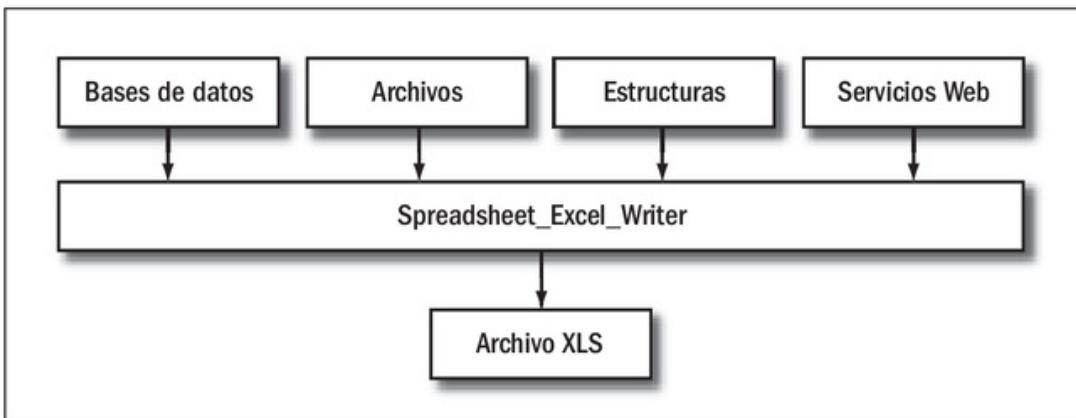


Figura 17. La generación de documentos XLS es independiente del origen de los datos manejados.

El estilo **fondo** asigna un color a las celdas definidas en **write** (que recibe como cuarto argumento opcional la referencia al estilo) utilizando el método **setBgColor**. Por supuesto, hay otros métodos disponibles relacionados con el diseño de celdas, entre los más utilizados por los desarrolladores contamos con:

MÉTODO	DEFINICIÓN
setAlign	Sirve para aplicar alineamientos horizontales (left , center , right , fill , justify , merge , equal_space) y verticales (top , vcenter , bottom , vjustify , vequal_space). Para aplicar ambos en un mismo estilo, debemos realizar dos llamadas al método.
setBgColor	Define el color de fondo de una celda.
setBold	Aplica el estilo bold a una celda y recibe como argumento opcional el nivel de oscuridad (0 para texto normal, 1 para negrita, y luego admite un rango entre 100 y 1000). El valor por defecto es 1.
setBorder	Ancho para el borde de una celda. Admite como valores posibles 1 (thin) y 2 (thick).
setBorderColor	Asigna el color del borde de una celda. Los colores se representan mediante cadenas de caracteres (black , white , red , green , blue , yellow , magenta y cyan) o mediante números enteros entre 8 y 63. Mediante setCustomColor podemos definir colores personalizados.
setBottom	Ancho para el borde inferior de una celda. Admite como valores posibles 1 (thin) y 2 (thick).
setBottomColor	Asigna el color del borde inferior de una celda.
setColor	Define el color del contenido de una celda.
setCustomColor	Recibe como argumentos el índice del color que queremos modificar y, sucesivamente,

MÉTODO	DEFINICIÓN
	los valores RGB (Red Green Blue, hacen referencia a la intensidad de cada color primario en el color resultante) para formar el nuevo.
setFgColor	Define el color de la capa superior del fondo de una celda.
setFontFamily	Define el tipo de fuente. Posibles valores: Times New Roman, Arial y Courier.
setHAlign	Alternativamente, al utilizar setAlign para alineamientos horizontales, podemos aprovechar este método que admite los mismos argumentos.
setItalic	Cambia a itálicas el contenido de una celda. No recibe argumentos.
setLeft	Ancho para el borde izquierdo de una celda. Admite como valores posibles 1 (thin) y 2 (thick).
setLeftColor	Asigna el color del borde izquierdo de una celda.
setMerge	Es un alias para setAlign con merge.
setNumFormat	Define el formato numérico de una celda (decimal, fecha, fecha y hora, etcétera). Más acerca de esto, en la documentación oficial de la clase.
setPattern	Utilizado en conjunto con setBgColor y setFgColor , nos permite uniformar el diseño de fondo de una celda con líneas horizontales, verticales, diagonales, y las posibles combinaciones.
setRight	Ancho para el borde derecho de una celda. Admite como valores posibles 1 (thin) y 2 (thick).
setRightColor	Asigna el color del borde derecho de una celda.
setShadow	Sombrea el contenido.
setSize	Define el tamaño del contenido de una celda. Recibe como argumento un número entero que representa el tamaño en pixeles (px).
setTextRotation	Posiciona la rotación del texto. Posibles valores son 0, 90, 270, y -1.
setTextWrap	Indica que el contenido de la celda deberá respetar el ancho máximo de ella.
setTop	Ancho para el borde superior de una celda. Admite como valores posibles 1 (thin) y 2 (thick).
setTopColor	Asigna el color del borde superior de una celda.
setUnderline	Define el subrayado de una celda. Admite como valores 1 (subrayado simple) y 2 (subrayado doble).
setVAlign	Alternativamente, al utilizar setAlign para alineamientos verticales, podemos usar este método que admite los mismos argumentos.

Tabla 7. Algunos métodos disponibles para definir formatos de celdas.

Estas opciones pueden utilizarse en conjunto para definir estilos complejos, como por ejemplo:

```
$titulo =& $excel->addFormat();
$titulo->setBgColor('red');
$titulo->setBold();
$titulo->setFgColor('blue');

$titulo->setItalic();
$texto =& $excel->addFormat();
```

```
$texto->setAlign('center');
$texto->setSize(14);
```

El método **setPattern** nos permite, en algunos casos, variar el fondo de un conjunto de celdas generando un entramado:

```
<?php

require_once 'Spreadsheet/Excel/Writer.php';
$excel = new Spreadsheet_Excel_Writer();

$hoja =& $excel->addWorksheet();

//los patrones validos van desde el 0 hasta el 18
for ($i=0; $i<=18; $i++) {
    $formato =& $excel->addFormat();
    $formato->setBgColor('blue');
    $formato->setFgColor('yellow');
    $formato->setPattern($i);
    $hoja->write($i, 1, " xxx ", $formato);
}

$excel->send('ejemplo.xls');
$excel->close();

?>
```

Dentro de las hojas de cálculo, las celdas pueden tomar distintos tipos de valores: numéricos, vacíos, fórmulas, textuales, etcétera, por lo que el método **write** puede ser complementado a partir de otros más específicos:



Es relevante saber que la primera clase publicada por PEAR fue la llamada **DB**, cuya función es la misma que la que encontramos en MDB2, o sea, se trata de abstraer el acceso a bases de datos. Desde el sitio oficial del repositorio, se la considera obsoleta, pero de igual manera sigue disponible para los que quieran utilizarla en sus desarrollos.

MÉTODO	DEFINICIÓN
writeBlank	Escribe celdas en blanco. Los argumentos son el número de fila, el número de columna y el formato dado.
writeFormula	Sirve para escribir fórmulas, y los argumentos son el número de fila, el número de columna, la fórmula, y el formato dado. Las fórmulas comienzan con un signo de igualdad (=), y los argumentos están separados por comas, por ejemplo =SUM(A1,B1).
writeNote	Define un formato de tipo nota y recibe los mismos argumentos que write. No admite la inclusión de formato.
writeNumber	Define un formato numérico y recibe los mismos argumentos que write.
writeString	Define un formato de cadena de caracteres y recibe los mismos argumentos que write.
writeUrl	Imprime un hipervínculo y recibe como argumentos el número de fila, el de columna, el enlace, opcionalmente el texto que se mostrará en lugar del enlace, y un formato válido.

Tabla 8. Métodos disponibles para escribir valores en las celdas.

En el caso del método **writeFormula**, podría ser oportuno el uso del método **rowcolToCell**, que recibe como argumentos el número de fila y el de columna, devolviendo la dirección en la forma letra/número (**B2**, **A4**, etcétera):

```
$excel = new Spreadsheet_Excel_Writer();
echo $excel->rowcolToCell(3, 12);
```

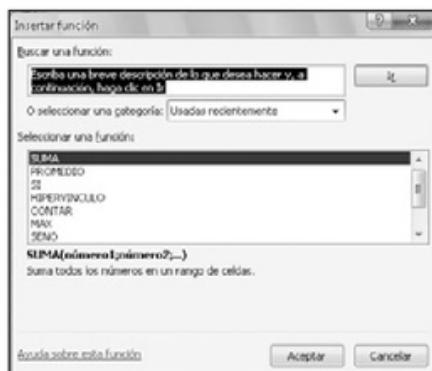


Figura 18. La inclusión de fórmulas es una de las opciones más utilizadas por los usuarios de hojas de cálculo.

III XML

Las últimas versiones de la aplicación de ofimática Microsoft Excel, (versiones 2007 y 2010) utilizan **XML** (*eXtensive Markup Language*) como formato interno para construir las hojas de cálculo, algo que sin dudas facilita la posibilidad de que otras aplicaciones puedan generar documentos compatibles sin mayores complicaciones.

Las hojas contenidas en un documento poseen sus propios métodos asociados. Además de los ya vistos, podemos citar a:

MÉTODO	DEFINICIÓN
activate	Marca una hoja como activa.
centerHorizontally	Centra la hoja horizontalmente.
centerVertically	Centra la hoja verticalmente.
freezePanes	Mantiene fijas ciertas regiones de la hoja.
getName	Devuelve el nombre de la hoja actual.
insertBitmap	Inserta una imagen (mapa de bits de 24bits) y recibe como argumentos la fila, la columna, el nombre de la imagen, el offset horizontal (opcional) dentro de la celda, el vertical (opcional) dentro de la celda, la escala horizontal (opcional) y la escala vertical (opcional).
mergeCells	Une la región dada como argumento (la primera fila, la primera columna, la última fila y la última columna).
printArea	Define el área de impresión de la hoja. Recibe como argumentos la primera fila, la primera columna, la última fila y la última columna.
protect	Marca la hoja como protegida. Recibe como argumento una contraseña.
select	Marca una hoja como seleccionada.
setColumn	Define el ancho de un rango de columnas (el primer argumento es la primera y el segundo la última). El tercer argumento es el ancho en pixeles, y el cuarto un formato (opcional).
setFirstSheet	Marca una hoja como la primera.
setFooter	Define el pie de página y, opcionalmente, el margen requerido.
setHeader	Define el encabezado y, en forma opcional, el margen requerido (primer y segundo argumento respectivamente)
setHPagebreaks	Saltos de página horizontales. Recibe como argumento un array que contiene los números de filas.
setInputEncoding	Define el juego de caracteres de la hoja. Posibles valores son UTF-8 y ISO-8859-7.
setLandscape	Orientala hoja a lo largo.
setMarginBottom	Define un margen inferior a la hoja en pulgadas.
setMarginLeft	Define un margen izquierdo a la hoja en pulgadas.
setMarginRight	Define un margen derecho a la hoja en pulgadas.
setMargins	Define un margen a la hoja en pulgadas.

III INDEPENDENCIA

Debemos señalar que aun cuando la aplicación más popular para trabajar con hojas de cálculo sea Microsoft Excel proporcionado por los desarrolladores de Windows, los documentos generados con Spreadsheet_Excel_Writer no están ligados a una herramienta en particular, manteniendo la compatibilidad con otras opciones disponibles en el mercado.

MÉTODO	DEFINICIÓN
setMargins_LR	Define un margen izquierdo y derecho a la hoja en pulgadas.
setMargins_TB	Define un margen superior e inferior a la hoja en pulgadas.
setMarginTop	Define un margen superior a la hoja en pulgadas.
setPaper	Define el tipo de papel mediante un número entero. Podemos obtener un listado de los códigos disponibles en la dirección http://rockssolidolutions.org/reference/excel_file_format_paper_size_table.html .
setPortrait	Orientala hoja a lo alto.
setPrintScale	Define la escala por defecto de la hoja. El argumento es un entero que representa el porcentaje.
setRow	Define el alto de una fila. Recibe como argumentos el número de fila, el alto y, opcionalmente, un formato asociado.
setSelection	Elige cuáles celdas estarán seleccionadas. Recibe como argumentos la primera fila, la primera columna, la última fila y la última columna.
setVPagebreaks	Saltos de página verticales. Recibe como argumento un array que contiene los números de columnas.
setZoom	Define el zoom por defecto de la hoja. El argumento es un entero que representa el porcentaje.
thawPanes	Lo opuesto a freezePanes; deja de mantener fijas las regiones definidas.

Tabla 9. Métodos disponibles para las hojas de un documento.

En el siguiente ejemplo listamos un conjunto de películas, utilizando como fuente de datos un array PHP:

```
<?php

$peliculas[] = array('The Godfather (1972)', '9.2');
$peliculas[] = array('The Godfather: Part II (1974)', '9.1');
$peliculas[] = array('Rear Window (1954)', '8.7');
$peliculas[] = array('Raiders of the Lost Ark (1981)', '8.7');
$peliculas[] = array('The Usual Suspects (1995)', '8.7');
$peliculas[] = array('Psycho (1960)', '8.7');
$peliculas[] = array('North by Northwest (1959)', '8.6');
$peliculas[] = array('The Silence of the Lambs (1991)', '8.6');
$peliculas[] = array('Memento (2000)', '8.6');
$peliculas[] = array('Fight Club (1999)', '8.6');

require_once 'Spreadsheet/Excel/Writer.php';

$excel = new Spreadsheet_Excel_Writer();
$hoja =& $excel->addWorksheet('Películas');
```

```

$Texto =& $excel->addFormat();

$Texto->setAlign('center');

$hoja->setColumn(0, 0, 40);
$hoja->setColumn(0, 1, 10);

$hoja->writeString($fila, 0, 'Titulo');
$hoja->writeString($fila, 1, 'Puntaje', $Texto);
foreach ($peliculas as $pelicula) {

    $hoja->writeString(++$fila, 0, $pelicula[0]);
    $hoja->writeNumber($fila, 1, $pelicula[1], $Texto);
}

$excel->send('Peliculas.xls');
$excel->close();

?>

```



Figura 19. A través de *Spreadsheet_Excel_Writer*, podemos enviar la hoja de cálculo generada directamente al navegador, forzando la descarga.

{ } PERL

El proyecto denominado *Spreadsheet_Excel_Writer* surgió originalmente como una implementación de un módulo similar disponible en el lenguaje de programación Perl (que se caracteriza por contar con una gran variedad de opciones para trabajar con textos en los más diversos formatos) conocido como **Spreadsheet::WriteExcel**.

Formularios con HTML_QuickForm

HTML_QuickForm es un paquete que nos permite generar formularios HTML, asignar controles (**select**, **text**, **password**, **checkbox**, **file**, **submit**, **reset**, **button**, **image**, **radio**, **hidden**, **textarea**, etcétera) y realizar validaciones tanto en el lado cliente (con JavaScript) como en el servidor (con PHP).

```
pear install HTML_Common
```

```
pear install HTML_QuickForm
```

HTML_QuickForm2 está en proceso de desarrollo y, por motivos de compatibilidad, utilizaremos **HTML_QuickForm**, que sirve para todas las versiones del lenguaje. La versión 2 (sólo para PHP versión 5) se instala de la siguiente manera:

```
pear install HTML_Common2
pear install HTML_QuickForm2
```

Para crear y mostrar un formulario, debemos incluir el archivo **QuickForm.php**:

```
require_once 'HTML/QuickForm.php';
```

El constructor **HTML_QuickForm** recibe los siguientes argumentos, todos opcionales:

- Identificador (valores de los atributos **name** y **id**).
- Método (**GET** o **POST**).
- Acción (valor del atributo **action**).
- Target (**_BLANK**, **_SELF**, etcétera).
- Array de atributos extras para el elemento.
- Campo especial de verificación (**hidden**) para controlar el envío (puede ser verdadero o falso).

```
$form = new HTML_QuickForm('nombreForm', 'POST', '/index.php', NULL, NULL,
true);
```

```
$form->display();
```

Lo anterior genera un código como el siguiente:

```
<form action="/index.php" method="post" name="nombreForm" id="nombreForm">

<input name="_qf_nombreForm" type="hidden" value="" />
</form>
```

Abarcar todas las opciones y funcionalidades brindadas por este paquete es poco menos que imposible: el manual oficial de PEAR incluye un detalle exhaustivo acerca de cada método disponible, por lo que recomendamos seguir avanzando sobre las características puntuales de HTML_QuickForm por este medio y por los diversos artículos publicados en los sitios especializados.

The screenshot shows a web browser displaying the PEAR Documentation website. The URL is <http://pear.php.net/manual/en/html.form.php>. The page title is "HTML_Form". The main content area contains sections like "Creating a form", "Creating the form in 'table mode'", and "Usage examples". It includes sample PHP code for creating a form. On the right side, there is a sidebar with links to "PEAR Manual", "HTML_Form", "Introduction", "Usage examples", "Package Info", and "API Documentation". The top navigation bar includes links for Main, Support, Documentation, Packages, Package Proposals, Developers, Bugs, and About PEAR. There is also a search bar at the top.

Figura 20. El generador de formularios de PEAR cuenta con un gran número de métodos divididos en categorías, como elementos HTML, controles personalizados, formato de salida, etcétera.

Agregar elementos

Para agregar elementos, contamos con el método **addElement**, que recibe como argumentos el tipo de control, un nombre (especificado en el atributo **name**), y un título por defecto para incluir en él:

```
$form->addElement('text', 'nombreControl', 'Escriba aqui su nombre:');
```

Que imprime como resultado:

```
<table border="0">
<tr>
  <td align="right" valign="top"><b>Escriba aqui su nombre: </b></td>
  <td valign="top" align="left"><input name="nombreControl" type="text">
</td>
</tr>
</table>
```

Entre los tipos de elementos disponibles podemos citar los siguientes:

NOMBRE	DEFINICIÓN
advcheckbox	Para pasar todos los valores de los checkbox, no sólo los seleccionados.
autocomplete	Cuadro de texto con la opción de autocompletar. Utiliza JavaScript.
button	Botón.
checkbox	Cuadros de selección.
date	Ingreso de fechas y horas.
file	Archivos.
group	Agrupación de elementos de un formulario.
header	Título para el formulario.
hidden	Campos ocultos.
hiddenselect	Genera un select más campos ocultos con sus valores seleccionados.
hierselect	Genera selects maestro/detalle.
image	Imágenes.
link	Genera enlaces (hipervínculos).
password	Para contraseñas.
radio	Grupos de selección.
reset	Para volver el formulario a su situación inicial.
select	Selección de opciones.
static	Datos estáticos.
submit	Botón para enviar el formulario.
text	Cuadro de texto.
textarea	Cuadro de texto de múltiples líneas.
xbutton	Botón.

Tabla 10. Elementos disponibles soportados.

Todos los argumentos a **addElement** a partir del segundo, son opcionales.

Conoceremos más acerca de los argumentos particulares de los elementos disponibles, en la sección dedicada a la clase HTML_QuickForm del manual de PEAR.

```
$array = array('rows'=>'10', 'cols'=>'5');
$form->addElement('textarea', 'ta', 'Descripcion', $array);
```

Por ejemplo, un control de tipo **select** admite como cuarto argumento las opciones disponibles:

```
$opciones[1] = 'a';
$opciones[2] = 'b';
$opciones[3] = 'c';

$form->addElement('select', 'nombre', 'label', $opciones);
```

Por defecto, cada llamada a **addElement** ubica el nuevo control en una nueva línea (fila de una tabla). Para ingresar elementos y mostrarlos uno detrás de otro, existe el método **addGroup**:

```
$grupo[] = $form->createElement('reset', 'reset', 'Limpiar');
$grupo[] = $form->createElement('submit', 'submit', 'Enviar');

$form->addGroup($grupo, $nombre, $titulo, $separador);
```

El método **createElement** es similar a **addElement** sólo que, en lugar de agregar un elemento al formulario, simplemente, lo asigna a una variable.

El método **setDefaults** nos permite asignar valores por defecto a los elementos del formulario. Recibe como argumento un array cuyas claves serán los nombres de los controles. Cada posición tomará los valores que se quieran establecer:

```
$valores['nombre'] = 'Karen';

$valores['apellido'] = 'O';

$form->addElement('text', 'nombre', 'Nombre');
$form->addElement('text', 'apellido', 'Apellido');

$form->setDefaults($valores);

$form->display();
```

Si en lugar de valores variables queremos asignar valores constantes, disponemos del método **setConstants**, que funciona de manera similar.

Validar formularios

Como veremos a continuación, HTML_QuickForm pone a nuestra disposición dos métodos, con ligeras diferencias, para validar la entrada de datos por parte de un usuario: **validate** y **addRule** antes de su proceso o descarte.

Para considerar como válido un formulario, primero debe ser enviado y, luego, superar las reglas definidas. En el siguiente ejemplo, no se aplican reglas, por lo que, al presionar el botón **Enviar**, el formulario se considera válido:

```
<?php

require_once "HTML/QuickForm.php";

$form = new HTML_QuickForm('form', 'post');
$form->addElement('header', 'titulo', 'Formulario de Prueba');
$form->addElement('text', 'texto', 'Ingrese un texto');
$form->addElement('reset', 'botonReset', 'Reset');
$form->addElement('submit', 'botonEnviar', 'Enviar');

if ($form->validate()) {
    echo 'El formulario ha sido enviado.';
} else {
    echo 'El formulario no ha sido enviado.';
}

$form->display();

?>
```

III REGLAS

El método **addRule** nos brinda algunos tipos de validaciones predefinidas y, también, admite la posibilidad de que desarrollemos las nuestras propias mediante el método **registerRule**. Éste recibe como argumentos un nombre para el tipo de validación (que luego se utilizará en **addRule**), el valor '**callback**' y el nombre de la función PHP.

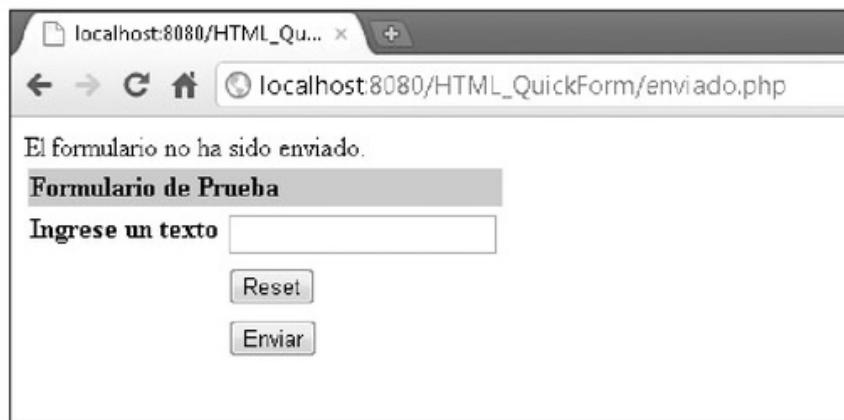


Figura 21. La utilización de *HTML_QuickForm* nos permite mantener un control profundo sobre el comportamiento de cada elemento.

Para definir reglas, contamos con el método **addRule**, que recibe como argumentos el nombre del control por validar, un mensaje para mostrar al usuario en caso de que la regla no haya sido superada y un tipo de validación:

```
$form->addRule('texto', 'Por favor complete este campo', 'required');
```

Los distintos tipos de validación pueden llegar a requerir un cuarto argumento al método **addRule**; entre los disponibles se encuentran:

NOMBRE	DESCRIPCIÓN	CUARTO ARGUMENTO A ADDRULE
alphanumeric	El valor sólo debe contener letras y/o números.	Ninguno.
callback	Pasa el control a una función externa.	Nombre de la función.
compare	Para utilizar esta regla, se pasa como argumento a addRule un array de elementos para realizar una comparación.	Define el tipo de comparación, puede ser: == (igualdad), != (distinto que), > (mayor), >= (mayor o igual), < (menor), <= (menor o igual).
email	Valida la correcta formación de una dirección de correo electrónico.	Verdadero para validar el dominio de la dirección mediante la función checkdnsrr.
filename	Aplica una expresión regular al nombre de archivo cargado.	Expresión regular.
lettersonly	El valor sólo debe contener letras.	Ninguno.
maxfilesize	Valida que el archivo cargado no supere un determinado número de bytes.	Número de bytes.

NOMBRE	DESCRIPCIÓN	CUARTO ARGUMENTO A ADDRULE
maxlength	Respetar la longitud máxima del campo.	Longitud máxima.
mimetype	Valida el tipo MIME del archivo cargado.	Una cadena o un array que contiene los valores permitidos.
minlength	Respetar la longitud mínima del campo.	Longitud mínima.
nonzero	Valor numérico que no comience con 0.	Ninguno.
nopunctuation	El valor no debe contener caracteres de puntuación.	Ninguno.
numeric	El valor sólo debe contener números.	Ninguno.
rangelength	El valor debe estar contenido entre un límite inferior y uno superior.	Array que contiene el límite inferior y el superior.
regex	El valor debe respetar una expresión regular.	Expresión regular.
required	El valor no debe ser vacío.	Ninguno.
uploadedfile	Valida que el archivo haya sido cargado. La regla required no funciona para archivos.	Ninguno.

Tabla 11. Tipos de validaciones soportadas.

Algunos ejemplos de aplicación de reglas de validación:

```
$form->addRule('direccion', '*', 'alphanumeric');

$form->addRule('edad', '*', 'numeric');

$form->addRule(array('pass1', 'pass2'), '*', 'compare', '==');

$form->addRule('mail', '*', 'email');

$form->addRule('apellido', '*', 'lettersonly');

$form->addRule('edad', '*', 'nopunctuation');

$form->addRule('puntaje', '*', 'rangelength', array(1,10));

$form->addRule('pass1', '*', 'required');

$form->addRule('imagen', '*', 'uploadedfile');
```

Alternativamente, podemos realizar la validación de las reglas en el lado cliente a través de JavaScript definiendo mediante un quinto parámetro opcional (**client**) al método **addRule**, esto será mucho más dinámico para el usuario:

```
$form->addRule('apellido', '*', 'required', '', 'client');
```

Esto no es válido, por ejemplo, para las validaciones vinculadas con la carga de archivos (**uploadedfile**, **maxfilesize**, etcétera). En cualquiera de los dos casos, debemos incluir las reglas antes de invocar al método **validate**:

```
<?php

require_once "HTML/QuickForm.php";

$form = new HTML_QuickForm('form', 'post');
$form->addElement('header', 'titulo', 'Formulario de Prueba');
$form->addElement('text', 'texto', 'Ingrese un texto');
$form->addElement('reset', 'botonReset', 'Reset');
$form->addElement('submit', 'botonEnviar', 'Enviar');
$form->addRule('texto', 'Por favor complete este campo', 'required');

if ($form->validate()) {

    header("location: gracias.html");
    exit;
}

$form->display();

?>
```

Es posible definir procesamientos sobre los valores de los campos antes de que éstos sean validados. El método **applyFilter** recibe como argumentos el nombre del campo y una función para aplicar sobre su valor:

III XHTML

Para los desarrolladores, es importante que más allá de las facilidades que nos brinda el conocido **HTML_QuickForm** para generar formularios HTML, un punto de gran importancia en muchos casos es que el código generado es compatible con XHTML (*eXtensible HyperText Markup Language*), lo cual implica que respeta en forma total sus reglas.

```
$form->addRule('apellido', '*', 'required');
$form->applyFilter('apellido', 'trim');
```

Otro método relacionado es **freeze**, que permite anular la posibilidad de editar los campos del formulario que han superado las reglas de validación, esto bloquea aquellos campos que el usuario completó correctamente:

```
<?php

require_once "HTML/QuickForm.php";

$form = new HTML_QuickForm('form', 'post');
$form->addElement('header', 'titulo', 'Formulario de Prueba');
$form->addElement('text', 'texto', 'Ingrese un texto');
$form->addElement('reset', 'botonReset', 'Reset');
$form->addElement('submit', 'botonEnviar', 'Enviar');

if ($form->validate()) {

    $form->freeze();
}

$form->display();

?>
```



Figura 22. La validación de datos en el servidor da más posibilidades al programador PHP, permitiendo realizar controles personalizados.

Enviar datos

Una vez validados los campos mediante reglas y filtros, podemos redirigir todas las variables y pasárlas como argumento a una función a través del método **process**:

```
<?php

require_once "HTML/QuickForm.php";

$form = new HTML_QuickForm('form', 'post');

$form->addElement('header', 'titulo', 'Formulario de Prueba');
$form->addElement('text', 'texto', 'Ingrese un texto');
$form->addElement('reset', 'botonReset', 'Reset');
$form->addElement('submit', 'botonEnviar', 'Enviar');

if ($form->validate()) {
    $form->freeze();
    $form->process('procesarDatos', false);
}

$form->display();

function procesarDatos ($datos) {
    echo "<pre>";
    print_r($datos);

    echo "</pre>";
}

?>
```

III CAPTCHA

Es habitual que encontremos imágenes CAPTCHA (*Completely Automated Public Turing test to tell Computers and Humans Apart*, Prueba de Turing pública y automática para diferenciar a máquinas y humanos) en los formularios. PEAR nos ofrece una opción para generar estos gráficos (**Text_Captcha**, que vimos en el **Capítulo 7**).

En el ejemplo, la función **procesarDatos** recibe como argumento un array que contiene los campos enviados por formulario y sus respectivos valores. El segundo argumento a **process** especifica si se han enviado archivos o no.



Figura 23. Mediante el método **process** es posible enviar un array con los datos del formulario de forma automática.

Salida sin tablas

Por defecto, HTML_QuickForm genera formularios tomando como estructura base tablas HTML. Para modificar este comportamiento y generar salidas **table-less** (es decir, sin la utilización de tablas) deberemos instalar el paquete **HTML_QuickForm_Renderer_Tableless** (que incluye métodos para administrar el diseño de las salidas) y utilizarlo en conjunto con HTML_QuickForm de la siguiente manera:

```

<?php

require_once 'HTML/QuickForm.php';
require_once 'HTML/QuickForm/Renderer/Tableless.php';

```

III TEMPLATES

Debemos tener en cuenta que uno de los puntos fuertes de PEAR es la manera en que los paquetes se estructuran, ya que se encargan de permitir al desarrollador utilizarlos en conjunto. Acerca de esto mismo, HTML_QuickForm puede ser utilizado junto con diversos motores de templates como por ejemplo **IT**, **Sigma** y **Flexy**, entre otros.

```
$form = new HTML_QuickForm('formulario', 'post');
$salida = new HTML_QuickForm_Renderer_Tableless();

//... código ...

$form->accept($salida);
echo $salida->toHtml();

?>
```

Otro paquete disponible es **HTML_QuickForm_DHTMLRulesTableless**, que permite validar formularios en el lado cliente no sólo ante el evento **onSubmit**, sino también ante **onBlur** y **onChange**:

```
<?php

require_once 'HTML/QuickForm.php';
require_once 'HTML/QuickForm/DHTMLRulesTableless.php';
require_once 'HTML/QuickForm/Renderer/Tableless.php';

$form = new HTML_QuickForm_DHTMLRulesTableless('formulario', 'post');
$salida = new HTML_QuickForm_Renderer_Tableless();

//... código ...

$form->getValidationScript();
$form->accept($salida);

echo $salida->toHtml();

?>
```

GRUPOS

El método denominado **addGroupRule** se encarga de permitirnos realizar la aplicación de reglas sobre grupos de elementos. Es posible definir también el número de elementos dentro del grupo que deben superar una regla para considerar sobrepasado el proceso de validación, lo que se define mediante el quinto argumento al método.

The screenshot shows a Wikipedia article titled "Tableless web design". The page header includes links for "New features", "Log in / create account", "Article", "Discussion", "Read", "Edit", "View history", and "Search". The main content discusses the philosophy of "Tableless web design" and its evolution from HTML tables to CSS. A sidebar on the right is titled "Cascading Style Sheets" and lists various related topics. The left sidebar contains a navigation menu with links like "Main page", "Contents", "Featured content", etc.

Figura 24. El diseño sin tablas es una técnica de gran consenso, y cada vez son más los sitios que la implementan.

Para obtener la información de última hora acerca de los paquetes disponibles a través de PEAR, podemos visitar su sitio oficial <http://pear.php.net>.

... RESUMEN

PEAR es el repositorio de clases oficial de PHP, y, en este capítulo, introducimos sus principales características: la instalación, actualización y uso de paquetes, las opciones disponibles desde la línea de comandos y las diversas utilidades que nos brinda. Para observar sólo algunas de las opciones disponibles, dimos un vistazo general acerca de tres paquetes de gran popularidad: MDB2 (abstracción para acceso a datos), Spreadsheet_Excel_Writer (documentos xls) y HTML_QuickForm (formularios HTML).



ACTIVIDADES

PREGUNTAS TEÓRICAS

- 1** ¿Cuál cree que es la principal ventaja de PEAR?
- 2** ¿Para qué sirve el código que se descarga desde <http://pear.php.net/go-pear>?
- 3** ¿Qué función cumple el administrador de paquetes?
- 4** ¿Para qué sirve el comando `pear list`?
- 5** ¿Qué importancia tiene la directiva `include_path` para PEAR?
- 6** ¿Para qué sirve la clase `PEAR_Error`?
- 7** ¿Para qué sirven los drivers del paquete MDB2?
- 8** ¿Qué permite el método `writeFormula` de `Spreadsheet_Excel_Writer`?
- 9** ¿Para qué sirve el método `addRule` de `HTML_QuickForm`?
- 10** ¿Para qué sirve el método `process` de `HTML_QuickForm`?

EJERCICIOS PRÁCTICOS

- 1** Desarrolle un formulario de contacto utilizando `HTML_QuickForm`.
- 2** Almacene información en una base de datos utilizando MDB2.
- 3** Exporte los datos guardados utilizando `Spreadsheet_Excel_Writer`.
- 4** Migré la aplicación a otro motor de base de datos.
- 5** Agregue más campos al formulario.

Generación y tratamiento de gráficos

La inclusión de gráficos en aplicaciones web es algo habitual desde hace un tiempo, y el número de librerías que nos permiten agilizar el proceso de generación y manipulación de imágenes crece día a día.

Aquí haremos un repaso por algunas de ellas, observando cómo funcionan y en qué aspectos pueden favorecer nuestros desarrollos.

La librería GD	316
PHP Thumbnailer	319
phpThumb	338
JpGraph	349
PEAR	378
Imagick	386
Resumen	389
Actividades	390

LA LIBRERÍA GD

PHP provee una extensión para utilizar la librería **GD** (www.boutell.com/gd) a través de funciones predeterminadas. Permite dinamizar las salidas valiéndose de los ingresos de los usuarios o de datos almacenados en bases de datos o archivos de configuración, por ejemplo.



Figura 1. GD es la librería para generación y tratamiento de gráficos más utilizada en PHP.

Para su correcto funcionamiento, GD requiere que contemos con PHP versión 4.3.2 o superiores (la librería viene incluida en la distribución estándar). Además, deberá estar habilitada en el archivo **php.ini**:

```
extension=php_gd2.dll
```

III DISPONIBILIDAD DE LA LIBRERÍA

Normalmente, los servicios de alojamiento web (hosting) dan soporte para la utilización de la librería GD, por lo que no deberíamos tener inconvenientes en cuanto a la migración de aplicaciones entre un servidor y otro. De todas maneras y para evitar futuros problemas, es conveniente asegurarnos antes de realizar la contratación.

Para comprobar si disponemos o no de la librería, tenemos varias opciones, como funciones de PHP, entre ellas encontramos:

- Utilizar la función **phpinfo**:

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

- Verificar que esté habilitada alguna de las funciones de extensión:

```
<?php

if (function_exists("gd_info")) {
    echo "GD esta disponible";
    echo '<pre>';
    print_r(gd_info());
    echo '</pre>';
} else {
    echo "GD no esta disponible !";
}

?>
```

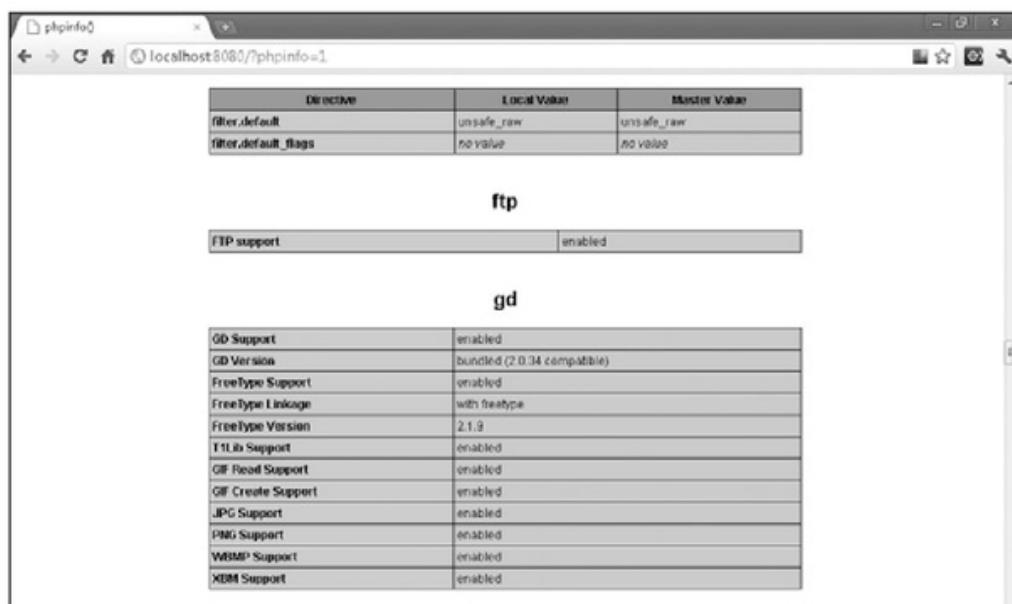


Figura 2. La función *phpinfo* nos permite conocer la versión de GD instalada en el servidor.

En instalaciones que requieren la compilación de PHP, deberán estar incluidas las siguientes opciones:

```
-with-gd
-with-jpeg-dir=/usr
-with-png
-with-ttf
```

Para verificar que nuestro sistema cuenta con estas alternativas habilitadas, nuevamente disponemos de la salida de la función **phpinfo**.

```
localhost:8080/gd/ view-source:localhost:8080/gd/
Array
(
    [GD Version] => bundled (2.0.34 compatible)
    [FreeType Support] => 1
    [FreeType Linkage] => with freetype
    [T1Lib Support] => 1
    [GIF Read Support] => 1
    [GIF Create Support] => 1
    [JPG Support] => 1
    [PNG Support] => 1
    [WBMP Support] => 1
    [XPM Support] =>
    [XBM Support] => 1
    [JIS-mapped Japanese Font Support] =>
)
```

Figura 3. La función *gd_info* devuelve información acerca del soporte brindado por nuestro sistema con relación a GD.

GD nos permite manipular diversos formatos de imágenes según su versión. Algunos de los formatos más populares soportados por GD son:

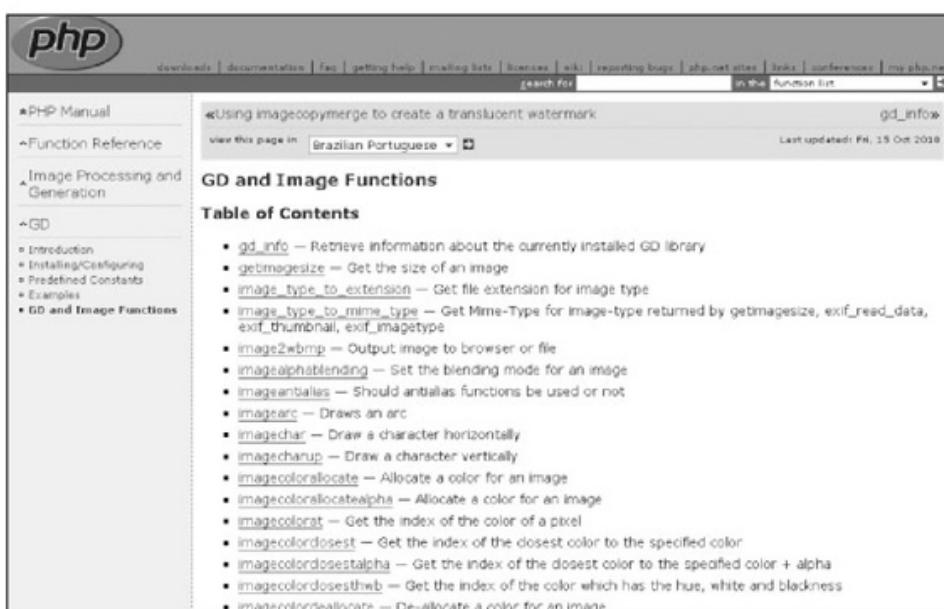


Figura 4. La extensión PHP para GD ofrece múltiples funciones para manipular gráficos en distintos formatos.

- **GIF** (*CompuServe Graphical Interchange Format*).
- **JPG** (o **JPEG**, *Joint Photographic Experts Group*).
- **PNG** (*Portable Network Graphics*, o *PNG is Not GIF*).

En las páginas siguientes, veremos algunas de las múltiples librerías escritas en PHP que hacen uso de GD para manipular imágenes.

PHP Thumbnailer

Una de las alternativas disponibles para manipular imágenes es **PHP Thumbnailer** (ahora rebautizada justamente como **PHP Thumb**, pero para evitar confusiones seguiremos llamándola **PHP Thumbnailer**), disponible para las versiones PHP5 y superiores. Podemos obtener más información acerca de sus características y modo de funcionamiento en el sitio web oficial: <http://phpthumb.gxdlabs.com>.

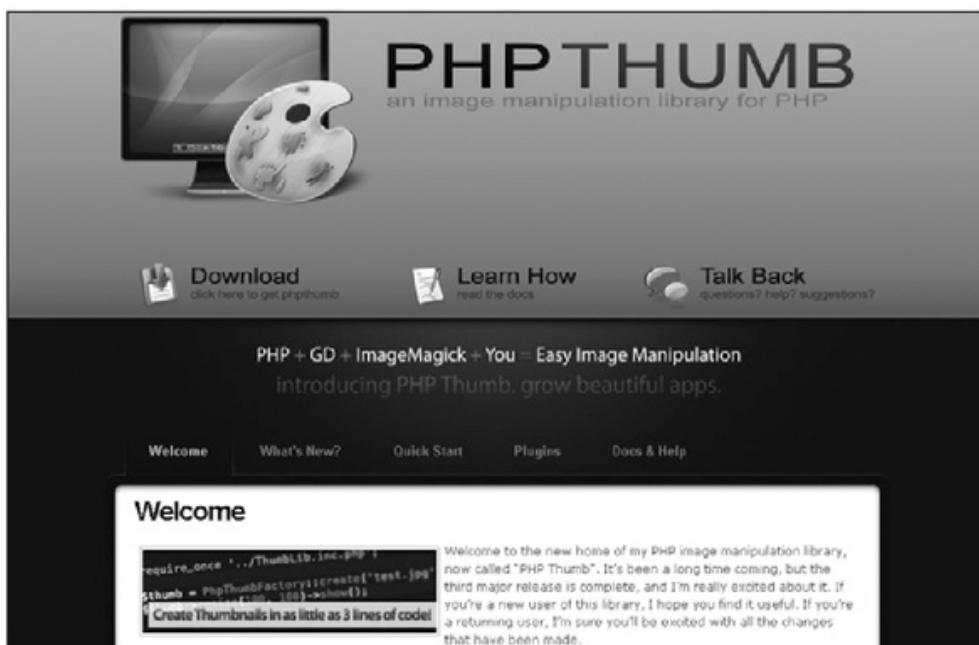


Figura 5. PHP Thumbnailer ofrece una interfaz simple y potente para manipular imágenes en distintos formatos.

Puede ser utilizada en conjunto tanto con **gd** (ésta es la opción por defecto) como con **imagick** (vista más adelante en este mismo capítulo).

Lo primero que deberemos hacer para comenzar a utilizar **PHP Thumbnailer** será incluir el archivo denominado **ThumbLib.inc.php**, el cual se encuentra junto a la distribución de la biblioteca:

```
require_once './Phpthumb/ThumbLib.inc.php';
```

PHP Thumbnailer es orientada a objetos (más información acerca de la **POO** en PHP en el **Capítulo 6**), por lo cual debemos instanciar la clase principal a través del método **create**, que recibe como argumento la ruta a la imagen:

```
$thumb = PhpThumbFactory::create('itc.png');
```

Este método admite como segundo argumento opcional un array con opciones, entre las cuales se encuentran:

- **resizeUp**: puede ser true o false y nos permite definir si queremos o no escalar una imagen según las dimensiones especificadas.
- **jpegQuality**: nivel de calidad para las imágenes JPEG, de 1 a 100.
- **correctPermissions**: corregir posibles inconvenientes relacionados a los permisos de las imágenes sobre las que trabajamos.
- **preserveAlpha**: preservar o no las transparencias en archivos PNG.
- **alphaMaskColor**: aplicar una máscara sobre la imagen. El color se define a través de un array con tres posiciones, y cada valor debe estar entre 0 y 255.
- **preserveTransparency**: preservar o no las transparencias en archivos GIF.
- **transparencyMaskColor**: similar a **alphaMaskColor**, sólo que transparente.

El método **show** nos permite enviar la imagen al navegador:

```
<?php  
  
require_once './PhpThumb/ThumbLib.inc.php';  
  
$thumb = PhpThumbFactory::create('itc.png');  
$thumb->show();  
  
?>
```

III CONTRIBUCIONES

Como la mayoría de las librerías profesionales, GD es un proyecto libre que avanza y se robustece a partir de la contribución de una amplia comunidad de usuarios que se encuentran esparcidos a través de todo el mundo. Éstos aportan soluciones y nuevas funcionalidades que son incorporadas de inmediato por parte de los responsables principales.



Figura 6. PHP Thumbnailer nos brinda opciones para trabajar con PHP versión 5 y superiores.

El método **resize** recibe dos argumentos, el ancho y el alto de la nueva imagen generada:

```
<?php  
  
require_once './PhpThumb/ThumbLib.inc.php';  
  
$thumb = PhpThumbFactory::create('itc.png');  
$thumb->resize(150, 150);  
$thumb->show();  
  
?>
```



Figura 7. PHP Thumbnailer es una librería orientada a objetos que permite generar y tratar imágenes.

La relación ancho/alto original se mantiene. En relación a esto, no es necesario definir los dos argumentos, con uno es suficiente:

index.php

```
<?php

for ($c=50;$c<=200;$c+=20) {
    echo "<img src='imagen.php?w=$c' />";
}

?>
```

imagen.php

```
<?php

require_once './PhpThumb/ThumbLib.inc.php';

$thumb = PhpThumbFactory::create('itc.png');
$thumb->resize($_GET[w]);
$thumb->show();

?>
```



Figura 8. La manipulación de imágenes a través de PHP Thumbnailer se realiza de manera transparente y personalizable.

Un método alternativo es **resizePercent**, que recibe como argumento opcional un porcentaje (100 para el tamaño original, 200 para duplicar, etcétera):

```
$thumb->resizePercent(40);
```



Figura 9. La librería PHP Thumbnailer permite enviar las imágenes generadas directamente al navegador.

El método **crop** nos permite tomar sólo una porción de la imagen original. Recibe como argumentos la coordenada inicial (primeros dos parámetros, X e Y), el ancho y el alto:

```
<?php

require_once './PhpThumb/ThumbLib.inc.php';

$thumb = PhpThumbFactory::create('itc.png');
$thumb->crop(50, 10, 210, 160);
$thumb->show();

?>
```



Figura 10. A través de PHP Thumbnailer es posible modificar las características de una imagen sin tocar el archivo original.

El método **cropFromCenter** es similar: recibe como argumentos el ancho y el alto, y toma como coordenada inicial el centro de la imagen:

```
<?php

require_once './PhpThumb/ThumbLib.inc.php';

$thumb = PhpThumbFactory::create('itc.png');
$thumb->cropFromCenter(200, 150);
$thumb->show();

?>
```

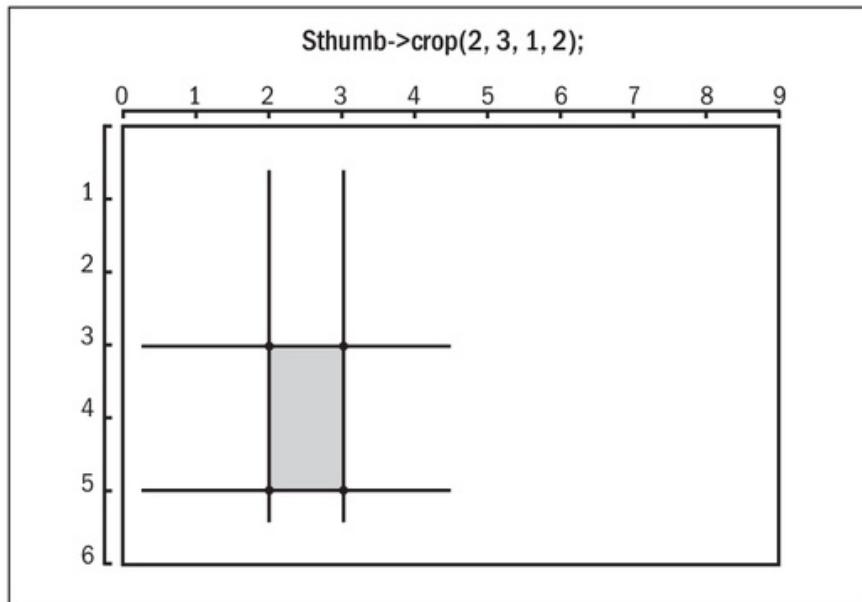


Figura 11. El recorte de imágenes a través del método *crop* se implementa a través de puntos y distancias.

En caso de recibir solo un argumento, asume como ancho / alto ese valor.

Un añadido interesante a esta nueva versión de **PHP Thumbnailer** es una combinación entre **resize** y **cropFromCenter**; es decir, definimos un alto / ancho y, si el aspect ratio no coincide, se hace un recorte a partir del centro. El método para lograrlo se denomina **adaptiveResize**:

```
<?php

require_once './PhpThumb/ThumbLib.inc.php';
```

```
$thumb = PhpThumbFactory::create('itc.png');
$thumb->adaptiveResize(300, 300);
$thumb->show();

?>
```



Figura 12. Las funcionalidades ofrecidas aumentan en cada versión, por lo cual recomendamos descargar la última disponible.

El método **rotateImage** rota una imagen 90 grados. De manera predeterminada, lo hace en el sentido de las agujas del reloj mediante el parámetro **CW**, pero para indicar lo opuesto, debemos utilizar **CCW**:

```
<?php

require_once './PhpThumb/ThumbLib.inc.php';

$thumb = PhpThumbFactory::create('itc.png');
$thumb->rotateImage(CCW);
$thumb->show();

?>
```



Figura 13. A través de PHP Thumbnailer es posible tomar partes específicas de una imagen definiendo las coordenadas correspondientes.

Más allá de estas constantes, podemos indicar números para representar los grados mediante el método `rotateImageNDegrees`:

```
$thumb->rotateImageNDegrees(180);
```



Figura 14. La rotación de imágenes es una muestra más de las características avanzadas de la librería.

III UTILIZACIÓN

Debemos saber que cada vez es más notoria la cantidad de aplicaciones basadas en GD que utilizan su potencial para la realización de tareas específicas. Esto habla de la estabilidad y de la seguridad que se ha alcanzado en el desarrollo de la librería en los últimos tiempos, algo que evidentemente genera confianza por parte de los desarrolladores.

Otra nueva mejora es la posibilidad de encadenar métodos:

```
<?php

require_once './PhpThumb/ThumbLib.inc.php';

$thumb = PhpThumbFactory::create('itc.png');
$thumb->rotateImage(CCW)->resize(150, 200)->cropFromCenter(100)->show();

?>
```



Figura 15. PHP Thumbnailer ofrece métodos para recuperar secciones particulares de una imagen.

El método **createReflection** permite incluir un reflejo de la imagen. Recibe como argumentos los siguientes:



Figura 16. Apple es una de las empresas que más utilizan la reflexión de imágenes en sus diseños.

- Porcentaje de la imagen por reflejar.
- El porcentaje correspondiente a la altura del reflejo generado (en relación con la altura original).
- Porcentaje de transparencia del reflejo.
- Incluir un borde a la imagen o no (de manera predeterminada se incluye).
- Color del borde (de manera predeterminada, #a4a4a4).

```
<?php

require_once './PhpThumb/ThumbLib.inc.php';

$thumb = PhpThumbFactory::create('itc.png');

$thumb->createReflection(50, 60, 90, true, '#a4a4a4');
$thumb->show();

?>
```



Figura 17. Las imágenes reflejadas son una de las características más salientes de la librería PHP Thumbnailer.

III

JAVASCRIPT

Mas alla de la gran variedad de opciones que nos entrega GD y los proyectos derivados, existe la posibilidad de aplicar efectos sobre imágenes utilizando librerías JavaScript. Script.aculo.us (<http://script.aculo.us>) y jQuery (<http://jquery.com>) son dos claros ejemplos que nos permiten ver con claridad la potencia de este lenguaje y la facilidad para implementar soluciones.

Para guardar alguna de las imágenes generadas (útil para crear miniaturas, por ejemplo), contamos con el método **save**, que recibe como argumentos el nombre del archivo (incluyendo la ruta) y, opcionalmente, el formato, que puede ser **png**, **jpg** o **gif**. Veamos un ejemplo:

```
<?php

require_once './PhpThumb/ThumbLib.inc.php';

$directorioImagenes = "./img";
$directorioMiniaturas = "./img/thumbs";

if ($handle = opendir($directorioImagenes)) {
    while ($file = readdir($handle)) {
        if (is_file("$directorioImagenes/$file")) {
            $imagenes[] = basename($file);

            $thumb = PhpThumbFactory::create("$directorioImagenes/$file");
            $thumb->adaptiveResize(150, 150);
            $thumb->save("$directorioMiniaturas/$file");
        }
    }
}

?>
```

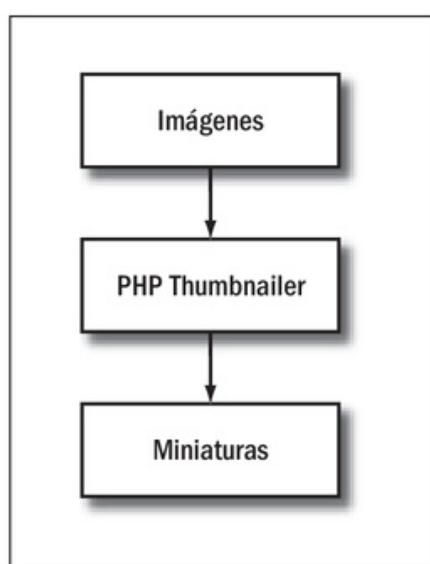


Figura 18. La generación de miniaturas es una necesidad habitual en los desarrollos actuales.

Veamos en una aplicación práctica cómo ampliar este ejemplo, generando en tiempo de ejecución miniaturas de imágenes. Para lograr este objetivo contaremos con un archivo para mostrar las imágenes generadas llamado **mostrarImagen.php**, el cual utilizaremos en breve:

```
<?php

// contenido de mostrarImagen.php

require_once './Phpthumb/ThumbLib.inc.php';

$thumb = PhpthumbFactory::create($_GET[archivo]);
$thumb->adaptiveResize($_GET[ancho], $_GET[alto]);
$thumb->show();

?>
```

Volviendo al index, primero definimos el directorio en donde están ubicadas las imágenes originales:

```
$directorioImagenes = "./img";
```

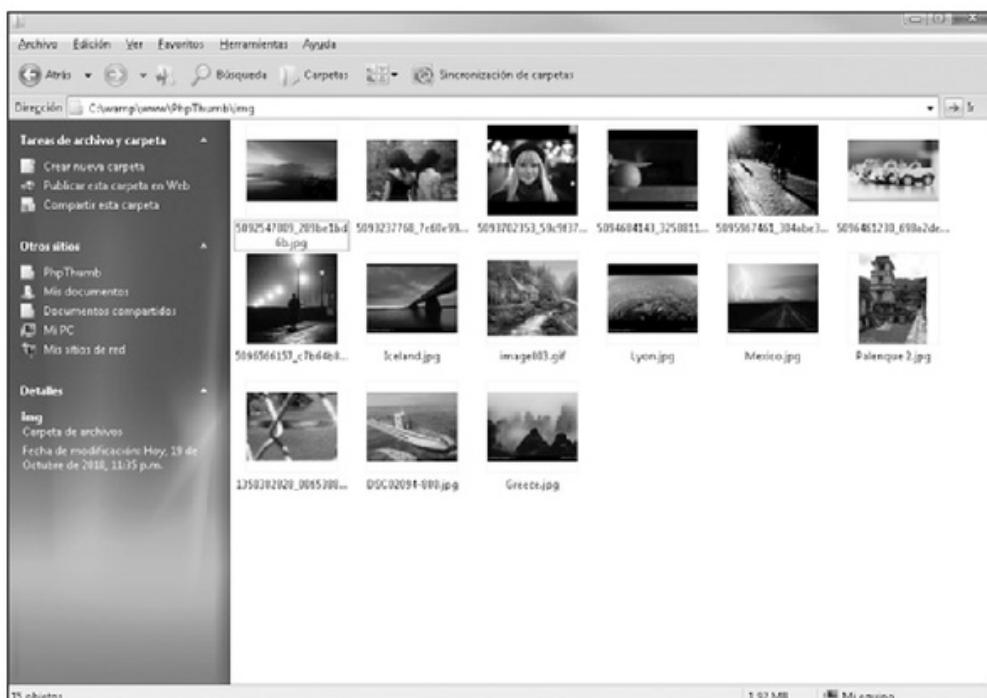


Figura 19. PHP Thumbnailer utiliza directorios temporales para generar imágenes a partir de otras.

Luego lo recorremos y cargamos los archivos en un array:

```
if ($handle = opendir($directorioImagenes)) {
    while ($file = readdir($handle)) {
        if (is_file("$directorioImagenes/$file")) {
            $imagenes[] = basename($file);
        }
    }
}
```

The screenshot shows the PHP Manual page for the `basename` function. The page has a sidebar on the left containing links to other PHP functions like `chgrp`, `chmod`, and `chown`. The main content area has a header "Filesystem Functions" and a sub-header "basename". It includes the PHP version information "(PHP 4, PHP 5)", a brief description ("Returns filename component of path"), and two sections: "Description" and "Parameters". The "Description" section contains the function signature `string basename (string $path [, string $suffix])` and a note about returning the base name of the file. The "Parameters" section details the `path` parameter as a path and the `suffix` parameter as an optional suffix to remove from the end of the path.

Figura 20. La función `basename` devuelve la parte de la ruta correspondiente al nombre del archivo.

III INTEGRAR LIBRERIAS

Una de las ventajas mas notables de PHP es la posilidad que nos da para incluir y combinar en un mismo proyecto multiples librerias para lograr desarrollos potentes y profesionales. Podremos darnos cuenta que la integracion no supone mayores inconvenientes y necesitaremos muy pocas lineas de código para aprovechar esta ventaja.

Para paginar los resultados obtenidos utilizaremos la clase **Pager** disponible a través de PEAR, que nos permite ordenar registros en páginas para navegarlas a través de enlaces. En el sitio oficial podemos encontrar más información al respecto, en el apartado dedicado a este paquete: <http://pear.php.net/package/PagerAdapter>. Primero incluimos el archivo principal **PagerAdapter.php**:

```
require_once 'PagerAdapter.php';
```

The screenshot shows the PEAR package information page for 'PagerAdapter'. At the top, there's a navigation bar with links for Main, Support, Documentation, Packages (which is selected), Package Proposals, Developers, and Bugs. Below the navigation is a search bar. The main content area has a title 'Package Information: Pager' and a subtitle 'Top Level :: HTML'. It features several tabs: Main, Download, Documentation, Bugs, and Trackbacks. Under the 'Main' tab, there are sections for 'Summary' (Data paging class: 'Data_paging'), 'Current Release' (version 2.4.8 released on 2009-04-21), 'Bug Summary' (listing 34 packages with open bugs, 2 total bugs, average age of 124 days, and oldest bug at 190 days), and 'Description' (explaining it takes an array of data and pages it according to various parameters). There's also a 'Report a new bug to Pager' link. The 'Maintainers' section lists Lorenzo Alberton and Richard Heyes. The 'More Information' section links to the external package homepage, source tree, RSS feed, and download statistics. At the bottom, there are links for 'Packages that depend on Pager' and 'Dependencies for Pager'.

Figura 21. El paquete *Pager* permite la paginación de estructuras PHP.

El constructor recibe como argumento un array que contiene diversas opciones de configuración (modo de navegación, registros por página, número de enlaces por página, datos por paginar, etcétera):

III APOYO

La extensión GD se incluye en las distribuciones oficiales de PHP desde hace ya algún tiempo, y es una de las pocas que se enfoca en la categoría de manipulación y generación de imágenes. Este hecho hace que los desarrolladores, en vista de la confianza por parte de PHP, le den una oportunidad y la utilicen en sus proyectos.

```
$parametros = array(
    'mode'      => 'Jumping', //o Sliding
    'perPage'   => 12,
    'delta'     => 4,
    'itemData'  => $imagenes
);

$paginador = & Pager::factory($parametros);
```

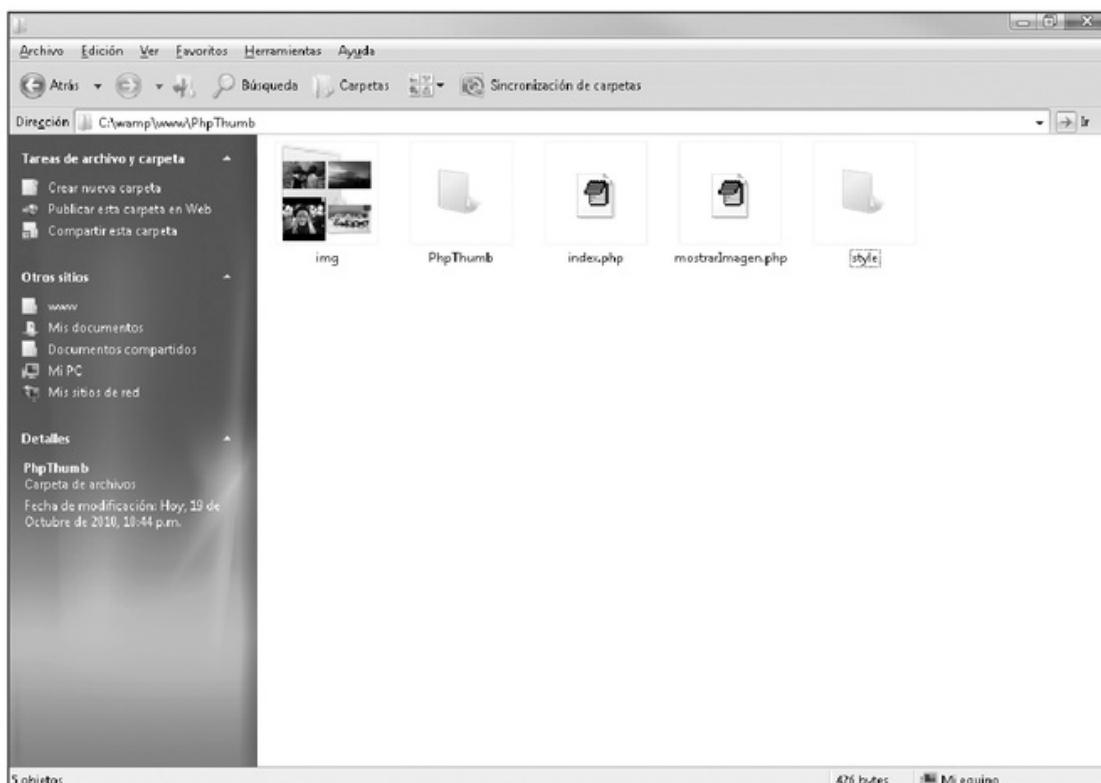


Figura 22. La utilización de PHP Thumbnailer nos da la posibilidad de generar múltiples vistas a partir de una misma imagen.

Los datos de la página actual (de manera predeterminada, la primera, si no especificamos lo contrario) se obtienen a través del método **getPageData**:

III PEAR

PEAR es un proyecto que cuenta con varios años en el mercado y que se utiliza en distintas clases de desarrollos, desde los mas simples hasta los mas complejos. Su integración no supone mayores inconvenientes y las soluciones que brinda son diversas, por lo que conviene consultar los paquetes disponibles para ver cuales nos pueden ser de utilidad en nuestros proyectos.

```
$registros = $paginador->getPageData();
```

Debemos tener en cuenta que si hay resultados disponibles, los recorremos y, con la ayuda del archivo que hemos denominado **mostrarImagen.php**, será posible que generemos las miniaturas correspondientes:

```
if (is_array($registros)) {
    echo '<table align="center">';
    echo '<tr>';
    foreach ($registros as $c => $v) {
        $contadorColumnas++;
        if ($contadorColumnas == 5) {
            $contadorColumnas = 1;
            echo '</tr><tr>';
        }

        $imagen = "<img
src='mostrarImagen.php?archivo=$directorioImagenes/$v&ancho=100&alto=100' />";
        echo '<td class="photo"><a href="img/' . $v . '">' . $imagen. '</a></td>';
    }
    echo '</tr>';
    echo '</table>';
}
```

Es importante saber que la barra de navegación se recupera mediante el uso del método denominado **getLinks**:

```
$links = $paginador->getLinks();
echo '<center><span class="paginador">' . $links['all'] . '</span></center>';
```



TAMAÑO DE IMAGENES

Una de las cuestiones básicas y principales a tener en cuenta en relación con las imágenes contenidas en un sitio es la necesidad de utilizar la menor cantidad posible de archivos y reducir el tamaño y peso de cada uno lo máximo que podamos. Siempre y cuando esto no afecte la correcta visualización, haciendo que nuestro sitio pierda calidad.



Figura 23. PHP Thumbnailer nos permite mantener el aspecto de las imágenes luego de redimensionarlas.

Otros métodos disponibles de **Pager** son:

MÉTODO	DESCRIPCIÓN
getCurrentPageID	Devuelve el número de página actual.
getNextPageID	Devuelve el número de página siguiente.
getPreviousPageID	Devuelve el número de página anterior.
numItems	Devuelve el número de registros disponibles.
numPages	Devuelve el número de páginas disponibles.
isFirstPage	Indica si la página actual es la primera.
isLastPage	Indica si la página actual es la última.
isLastPageComplete	Indica si la última página actual contiene numItems registros.
range	Devuelve un array numérico cuyas claves son las páginas actuales.

Tabla 1. Métodos puestos a disposición por la clase *Pager*.

III W3C

Es muy importante tener en cuenta que la organización conocida como W3C es la encargada de mantener un gran número de especificaciones relacionadas con diversas tecnologías abiertas, tiene en su sitio un apartado dedicado a PNG. Podemos visitarlo si accedemos a su sitio oficial, en www.w3.org/Graphics/PNG.



Figura 24. Las imágenes que utilizemos deberán ser válidas; de lo contrario, obtendremos un mensaje de error por parte de PHP Thumbnailer.

El contenido de **styles.css** es el siguiente:

```
.photo {
    padding:10px;
    border: 1px solid #cccccc;
    background: #E9ECEF;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: normal;
    font-size : 12px;
    text-align:center;
}

img {
    border:0px;
}

.paginador A {
    font-family: Verdana, Arial, Helvetica, Sans-serif;
    color: #666666;
    font-size: 10px;
    text-decoration: none;
}

.paginador {
```

```

font-family: Verdana, Arial, Helvetica, Sans-serif;
color: #000000;
font-size: 10px;
text-decoration: none;
}

```

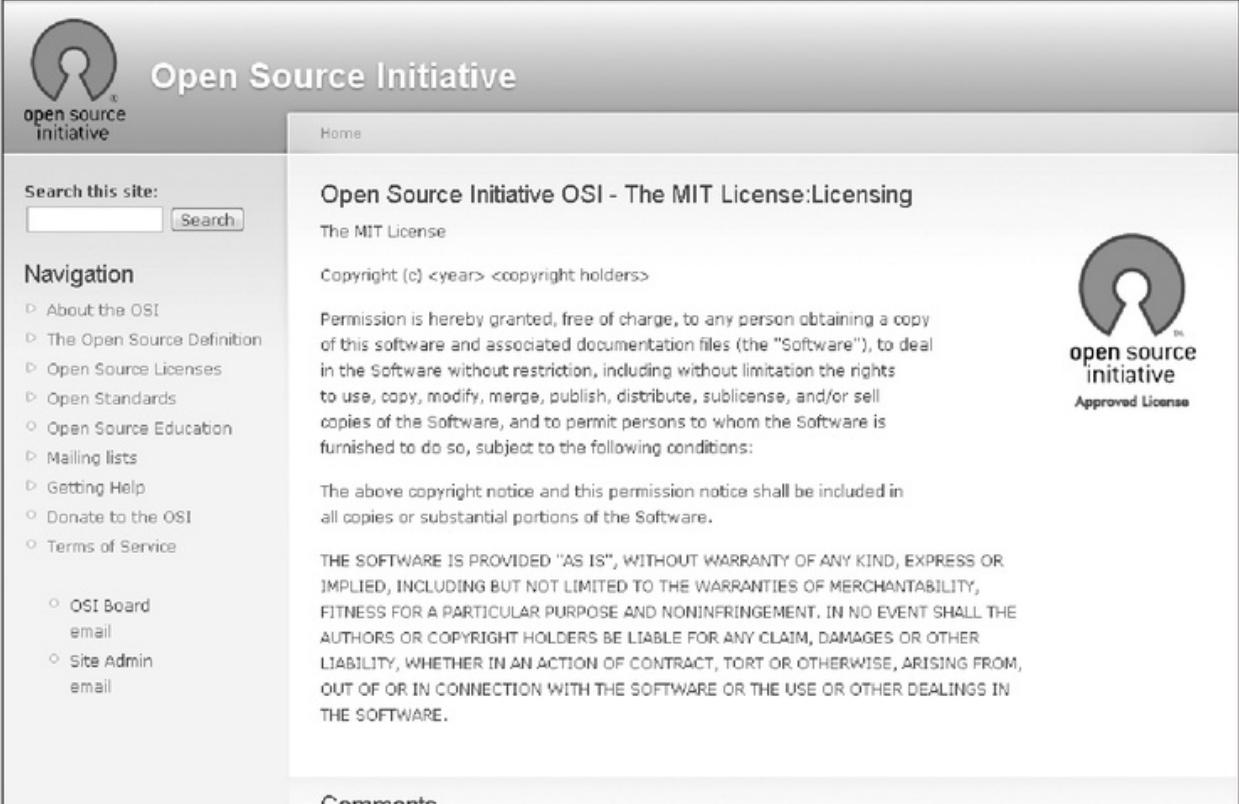


Figura 25. PHP Thumbnailer, al igual que otras librerías similares, permite devolver la imagen generada directamente al navegador.

Es importante tener en cuenta que PHP Thumbnailer es de uso libre y está disponible para los usuarios que lo deseen, bajo los términos indicados en la licencia conocida como **MIT**, acerca de la cual podemos obtener más información en www.opensource.org/licenses/mit-license.php.

III SVG

El organismo W3C ha desarrollado un formato de gráficos vectoriales llamado **SVG** (Scalable Vector Graphics). Aunque no es una novedad, está en pleno desarrollo y no son muchos los navegadores que lo soportan totalmente. Se trata de una alternativa para analizar y las posibilidades que obtenga dependerán de las empresas desarrolladoras de software.



The screenshot shows the Open Source Initiative (OSI) website. In the top left corner is the OSI logo, which is a stylized keyhole icon with the text "open source initiative". The main title "Open Source Initiative" is displayed prominently. On the left sidebar, there's a search bar with the placeholder "Search this site:" and a "Search" button. Below the search bar is a "Navigation" section containing links to various OSI resources like "About the OSI", "The Open Source Definition", and "Open Source Licenses". Further down the sidebar, there are links for the "OSI Board" and "Site Admin". The main content area is titled "Open Source Initiative OSI - The MIT License:Licensing". It includes sections for "The MIT License", "Copyright (c) <year> <copyright holders>", and "THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.". To the right of the main content, there's another instance of the OSI logo with the text "Approved License".

Figura 26. La licencia MIT otorga libertades a los usuarios para modificar y colaborar con los diferentes proyectos.

phpThumb

La librería **phpThumb** se utiliza para crear y manipular imágenes (**GIF**, **PNG** o **JPEG**) de forma dinámica. Aprovecha las características más avanzadas de la versión 2 de GD, aunque también es posible trabajar utilizando versiones anteriores. Trabaja con imágenes **GIF** aun cuando la versión de GD instalada en el servidor no soporte este formato (lo hace a través de la clase **GIFutil**). Además, mantiene un mecanismo para evitar el uso de la librería desde sitios externos o mostrar las imágenes almacenadas (**hotlinking**). Podemos conocer más en: <http://phpthumb.sourceforge.net>.

III REDIRECCIÓN DE LA SALIDA

El comportamiento predeterminado de la mayoría de las aplicaciones que se encargan de generar imágenes a partir de otras ya existentes consiste en enviar la salida directamente al navegador, por lo que debemos tener la precaución de no imprimir ninguna información antes, para no interferir en el procesamiento por parte del cliente.

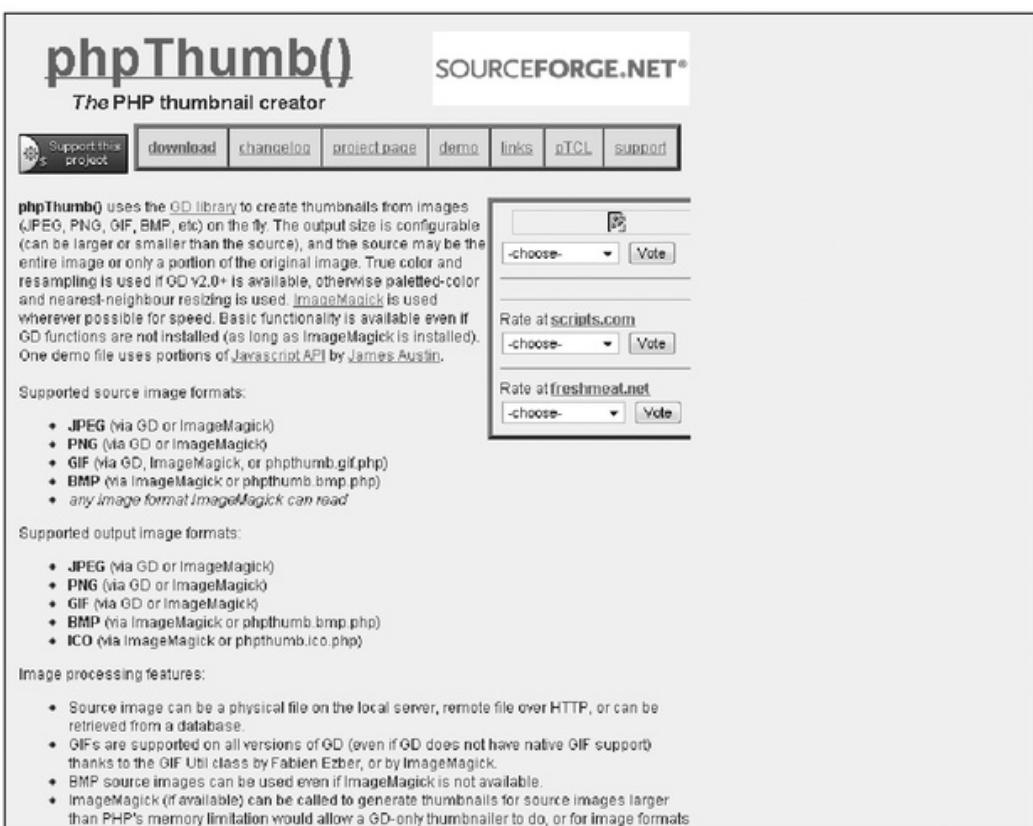


Figura 27. La librería **phpThumb** es un proyecto libre y muy utilizado en la actualidad.

Lo primero que debemos hacer luego de descargar la distribución y copiarla a un directorio del servidor será renombrar el archivo **phpThumb.config.php.default** a **phpThumb.config.php**. La forma de utilización de phpThumb es a través de una llamada al archivo **phpThumb.php**, que recibe una serie de parámetros y devuelve el resultado correspondiente, es decir, la imagen generada. El caso más común consiste en incluir la llamada dentro del atributo **src** del elemento **img**, como vemos en la siguiente etiqueta HTML:

```

```

Existe una sintaxis alternativa:

```
phpThumb.php/<parametro1>=<valor1>;<parametroN>=<valorN>;<w>x<h>;<imagen>
```

En donde el último argumento es la ruta a la imagen, el penúltimo son las dimensiones (ancho y alto) y lo demás son valores asignados a parámetros:

```

```

En el ejemplo anterior recuperamos la imagen **nombreImagen.png** y definimos que sea de 100px de ancho por 150px de alto. Además, incluimos dos parámetros adicionales: **f** (formato de la imagen devuelta) y **q** (calidad).



Figura 28. Dentro del archivo `phpThumb.config.php` hay disponibles múltiples directivas de configuración.

Entre los atributos más importantes puestos a disposición por phpThumb, se encuentran los siguientes:

- **src**: indica la dirección de la imagen por tratar.
- **new**: permite crear una imagen nueva y recibe como valores un número hexadecimal (color) y opcionalmente un porcentaje de opacidad. Además, requiere que se definan los atributos **w** y **h**. Con el siguiente código, generamos una imagen color amarillo de 200px por 100px:

```
phpthumb/thumb.php?new=FFFF00&w=200&h=100
```

Para definir el grado de opacidad (por ejemplo 20%):

```
phpthumb/thumb.php?new=FFFF00|20&w=200&h=100
```

- **w**: establece un ancho máximo en pixeles para la imagen.
- **h**: establece un alto máximo en pixeles para la imagen.
- **f**: establece un formato de salida para la imagen, que puede tomar como posibles valores a **jpeg**, **png** o **gif**.
- **q**: define la calidad (nivel de compresión) de la imagen y sólo se aplica al formato **jpeg** (1 para baja calidad, 95 para máxima calidad, 75 valor por defecto).
- **sx**: quita un porcentaje de la imagen a partir del margen izquierdo (toma valores entre 0 y 1). En el siguiente ejemplo, mostramos primero la imagen original y, luego, la misma imagen con un cuarenta por ciento menos:

```
phpthumb/phpthumb.php?src=/phpThumb/imagen2.png
phpthumb/phpthumb.php?src=/phpThumb/imagen2.png&sx=0.4
```



Figura 29. La manipulación de imágenes a través de phpThumb se realiza invocando al archivo *phpthumb.php*.

- **sy:** quita un porcentaje de la imagen a partir del margen superior (toma valores entre 0 y 1). En el siguiente ejemplo, mostramos primero la imagen original y, luego, la misma imagen con un diez por ciento menos. En la imagen podemos observar el resultado de esta función:

```
phpthumb/phpthumb.php?src=/phpThumb/imagen2.png
phpthumb/phpthumb.php?src=/phpThumb/imagen2.png&sy=0.1
```

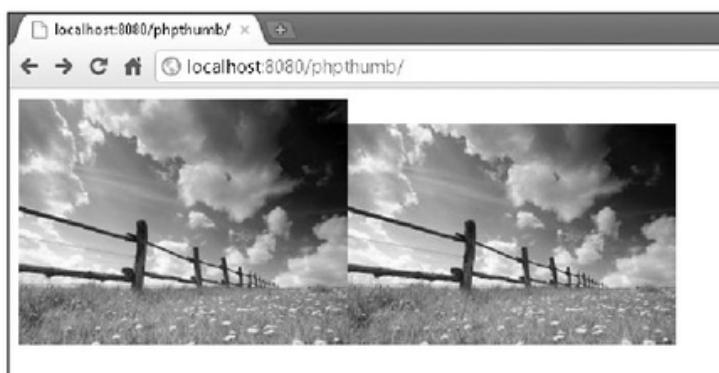


Figura 30. Con phpThumb es posible tomar sólo ciertos sectores de una imagen.

- **sw:** es similar a **sx** sólo que muestra únicamente el porcentaje a partir del margen izquierdo. En el siguiente ejemplo, mostramos primero la imagen original y, luego, el cincuenta por ciento de la misma imagen:

```
phpthumb/phpthumb.php?src=/phpThumb/imagen2.png
phpthumb/phpthumb.php?src=/phpThumb/imagen2.png&sw=0.5
```



Figura 31. *phpThumb trabaja con los diferentes formatos soportados por la librería GD.*

- **sh:** es similar a **sy** sólo que muestra únicamente el porcentaje a partir del margen superior. En el siguiente ejemplo, mostramos primero la imagen original y, luego, el setenta por ciento de la misma imagen. Observemos que si bien la sintaxis es similar, el resultado cambia notablemente:

```
phpthumb/phpthumb.php?src=/phpThumb/imagen2.png
phpthumb/phpthumb.php?src=/phpThumb/imagen2.png&sh=0.7
```

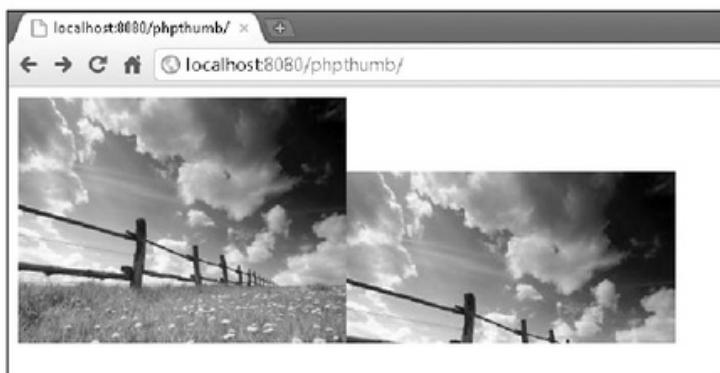


Figura 32. *La velocidad de procesamiento de phpThumb dependerá en parte del tamaño de las imágenes.*

- **zc:** devuelve una imagen con las dimensiones definidas cuyo contenido es la imagen original adecuada:

III VERSIONES

La librería conocida como **phpThumb** se encarga de trabajar con las distintas versiones de GD, sin embargo, debemos saber que su comportamiento puede cambiar según contemos con las últimas disponibles (desde la 2 en adelante) o las no tan nuevas. Algunas de las variaciones pueden estar relacionadas con la calidad en la visualización.

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&w=600&h=300&zc=1
```



Figura 33. La librería phpThumb permite mantener el aspecto de la imagen original.

- **bg:** permite modificar el color de fondo de una imagen (valor hexadecimal).
- **bc:** representa el borde de la imagen (valor hexadecimal).
- **ra:** rota la imagen de acuerdo con el ángulo dado (positivo, horario; negativo, antihorario):

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&ra=90
```

- **ar:** rota la imagen 90 grados:

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&w=200&ar=p  
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&w=200&ar=P
```



Figura 34. La rotación de imagen se puede realizar en sentido horario o antihorario.

- **iar** (*Ignore Aspect Ratio*): ignora las dimensiones originales de la imagen sin mantener su aspecto inicial.

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&w=420&h=120&iar=1
```



Figura 35. Las opciones que ofrece phpThumb se incluyen como argumentos al archivo *phpthumb.php*.

- **far:** crea una imagen con el ancho y alto especificados, pero mantiene el aspecto original, ubicándola adentro. El valor que recibe indica la alineación (**L** para izquierda, **R** para derecha, **T** para arriba, **B** para abajo, **C** para centro, y las variaciones posibles, por ejemplo, **BL** para abajo a la izquierda):

```

```



Figura 36. Es posible definir las dimensiones de una imagen sin perder el aspecto de la original.

- **sia** (*Save Image As*): define el nombre que se le dará a la imagen cuando el usuario quiera guardarla. No es necesario incluir la extensión:

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&sia=nombre
```

- **maxb**: nos da la posibilidad de establecer un valor máximo en bytes para la imagen. En el siguiente ejemplo definimos el peso máximo a 100000 bytes (alrededor de 100kb):

```
phpthumb/phpthumb.php?src=/phpthumb/imagen.jpg&maxb=100000
```

- **down**: permite forzar la descarga de una imagen, recibe como valor el nombre pre-determinado con el cual se guardará. La descarga comienza cuando accedemos a un enlace como el siguiente:

```
phpthumb.php?src=/phpthumb/imagen.jpg&f=png&down=nuevaImagen.png
```



Figura 37. Es posible configurar el nombre de la imagen percibido por el usuario de la aplicación.

A través del atributo **fltr**, phpThumb permite la aplicación de filtros sobre las imágenes. La forma de utilización es:

```
phpthumb.php?src=imagen.jpg&fltr[]=opcion|valor
```

El número de valores es variable:

```
phpthumb.php?src=imagen.jpg&fltr[]=opcion|valor1|valor2|valorN
```

Y podemos incluir más de un filtro a la vez:

```
phpthumb.php?src=imagen.jpg&fltr[]="opcion|valor&fltr[]="opcionN|valorN
```

Entre los filtros disponibles se encuentran:

NOMBRE	DESCRIPCIÓN
brit	Brillo, toma valores entre -255 y 255.
cont	Contraste, toma valores entre -255 y 255.
gam	Corrección Gama, toma valores positivos.
sat	Saturación, toma valores entre 0 y -100.
gray	Convierte a escala de grises, no recibe valores.
sep	Sepia, toma valores entre 0 y 100.
blur	Blur, toma valores entre 0 y 25.
over	Ubica una imagen por encima de otra. Recibe el nombre de la imagen y si se superpone a la original (0) o viceversa (1).
wmi	Agrega una marca de agua a la imagen. Recibe como valores la ruta a la imagen y la alineación (L para izquierda, R para derecha, T para arriba, B para abajo, C para centro, y las variaciones posibles, por ejemplo, BL para abajo a la izquierda).
wmt	Agrega un texto sobre la imagen. Recibe como valores el texto, el tamaño de la fuente (1 a 5), la alineación, y un color (valor hexadecimal), entre otras opciones.
flip	Da vuelta la imagen en sentido horizontal (valor x) o vertical (valor y).
ric	Redondea los bordes de la imagen. Recibe como valores el radio horizontal y el vertical.
bord	Asigna un borde a la imagen. Recibe como valores el ancho en pixeles, el radio horizontal y el vertical, y el color (hexadecimal).
crop	Extrae una parte de la imagen dadas las coordenadas (izquierda, derecha, arriba, y abajo; si están entre 0 y 1 se consideran porcentajes).

Tabla 2. Algunas de las opciones disponibles para el atributo **fltr**.

Como vimos, el archivo **phpthumb.php** admite una gran cantidad de argumentos, de los cuales el más importante es **src** que sirve para indicar la ruta a la imagen

III FINES

Es relevante recordar que una de las dudas más frecuentes a la hora de analizar una aplicación web es la de cómo mostrar la información generada a los usuarios que la necesiten. JpGraph es una aplicación que aparece como una valiosa opción en este contexto, ya que permite, de manera sencilla, generar gráficos intuitivos para exponer los datos almacenados.

(también es posible invocar archivos remotos, aunque puede ralentizar la normal ejecución del script dependiendo de su ubicación).

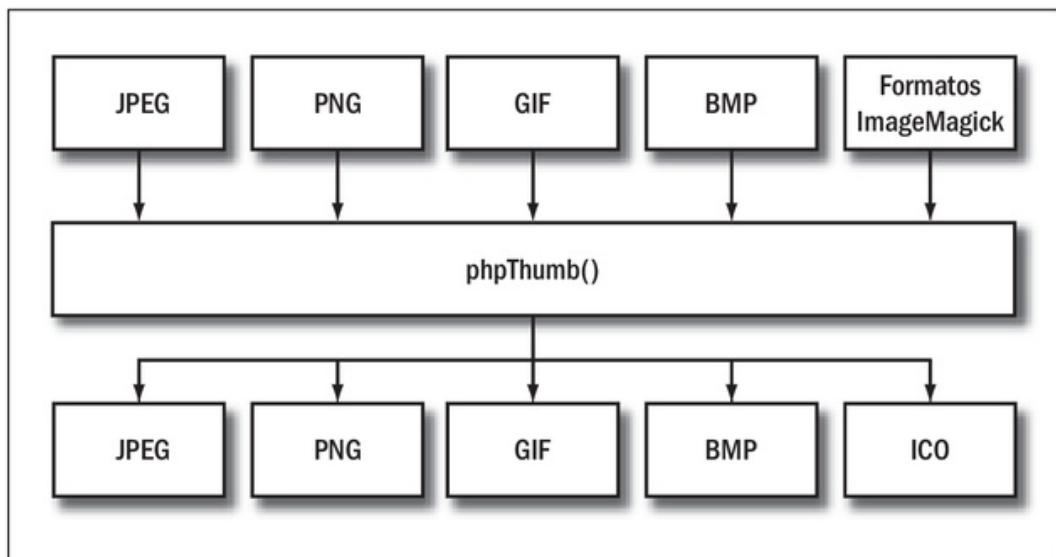


Figura 38. La librería *phpThumb* permite importar y exportar archivos de imagen en diferentes formatos.

phpThumb mantiene un avanzado sistema de caché que permite reutilizar las imágenes almacenadas en memoria para evitar procesamientos en el servidor. En el archivo **phpThumb.config.php**, se incluyen ciertas directivas para configurar el funcionamiento de la librería, entre las que se encuentra **cache_directory** (directorio de la caché):

```
$PHPTHUMB_CONFIG['cache_directory']
```

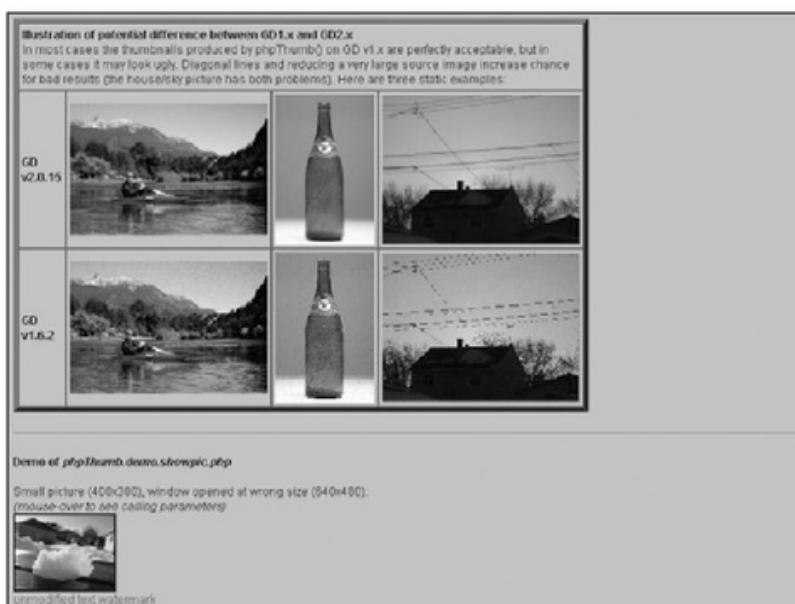


Figura 39. *phpThumb* permite trabajar utilizando cualquiera de las versiones de *GD* disponibles instaladas en el sistema.

Existen otras opciones relativas para configurar el comportamiento de la librería manteniendo un control exhaustivo sobre cada detalle personalizable a través del archivo **phpThumb.config.php** (más precisamente por medio del array **\$PHPTHUMB_CONFIG**), entre las cuales podemos citar las siguientes:

- Deshabilitar los mensajes de advertencia ante posibles fallos durante la generación o administración de la caché:

```
$PHPTHUMB_CONFIG['cache_disable_warning']
```

- Número de subdirectorios admitidos para organizar y almacenar los archivos de la caché:

```
$PHPTHUMB_CONFIG['cache_directory_depth']
```

- Eliminar imágenes almacenadas en caché que no hayan sido utilizadas en los **X** días posteriores al último acceso (**null** para nunca eliminar):

```
$PHPTHUMB_CONFIG['cache_maxage'] = 86400 * X;
```

- Tamaño máximo de la caché (en bytes):

```
$PHPTHUMB_CONFIG['cache_maxsize']
```

- Número máximo de archivos en la caché:

```
$PHPTHUMB_CONFIG['cache_maxfiles']
```

III EXPORTACIÓN DE GRÁFICOS

Como veremos, los gráficos generados a partir de JpGraph son, en definitiva, archivos de formatos conocidos y populares, y pueden ser incluidos en documentos **PDF** (PHP brinda librerías especiales para crear esta clase de archivos), por ejemplo, para generar reportes profesionales y personalizados por el propio desarrollador de la aplicación.

Translations of this page | Accessibility

 **GNU Operating System**

Sign up for the Free Software Supporter
A monthly update on GNU and the FSF

Philosophy Licenses Downloads Documentation Help GNU Join the FSF [Why GNU/Linux?](#)

GNU General Public License

- [A Quick Guide to GPLv3](#)
- [Why Upgrade to GPLv3](#)
- [Frequently Asked Questions about the GNU licenses](#)
- [How to use GNU licenses for your own software](#)
- [Translations of the GPL](#)
- The GPL in other formats: [plain text](#), [Texinfo](#), [LaTeX](#), [standalone HTML](#), [ODE](#), [Docbook](#)
- [GPLv3 logos to use with your project](#)
- [Old versions of the GNU GPL](#)
- [What to do if you see a possible GPL violation](#)



GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <<http://fsf.org/>>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Figura 40. *phpThumb* está disponible a través de la licencia **GNU GPL** (www.gnu.org/copyleft/gpl.html).

- Directorio temporal (**null** para autodetectar y utilizar el directorio temporal del sistema):

```
$PHPTHUMB_CONFIG['temp_directory']
```

- Formato de salida predeterminado (**jpeg**, **png** o **gif**):

```
$PHPTHUMB_CONFIG['output_format']
```

JpGraph

JpGraph es una librería orientada a objetos que permite la generación de gráficos, casi siempre de tipo estadístico. La cantidad de opciones de configuración relativas a la forma de visualizar de gráficos es inmensa y se detallan de manera clara y precisa en el manual de la aplicación, que incluye centenares de ejemplos prácticos de uso. Está disponible para todas las versiones de PHP y requiere que se habilite la extensión GD, cuyo objetivo es, como hemos visto, la generación de imágenes (la versión instalada en el servidor puede ir desde la 1.8 hasta la 2 y superiores).

Puesto que se trata de una herramienta en continuo desarrollo, en el sitio web oficial podremos encontrar información de última hora acerca de las mejoras introducidas.



Figura 41. JpGraph incluye distribuciones para las distintas versiones disponibles de PHP.

Al momento de descargar JpGraph, deberemos optar por una de las dos distribuciones: la definida para PHP 4 o la disponible para PHP 5 y superiores). Una vez descargada la librería, descomprimimos el archivo correspondiente y copiamos el directorio **src** (que contiene las clases) a un lugar accesible por las demás páginas. Para utilizar JpGraph, tenemos que incluir en nuestras páginas el archivo **jpgraph.php**:

```
include "jpgraph/jpgraph.php";
```

Luego, debemos agregar referencias a archivos que contienen las clases correspondientes al tipo de gráfico elegido, como veremos a continuación. Entre los tipos de gráficos soportados es posible mencionar:

- Lineales
- Barra
- Torta
- Anillo
- Mapas HTML
- Gráficos Polares
- Radar
- Diagramas de Gantt
- Texto (por ejemplo para generar imágenes CAPTCHA)
- Figuras
- Leds
- Código de barras

- Rosa de los vientos
- Velocímetros
- Puntos

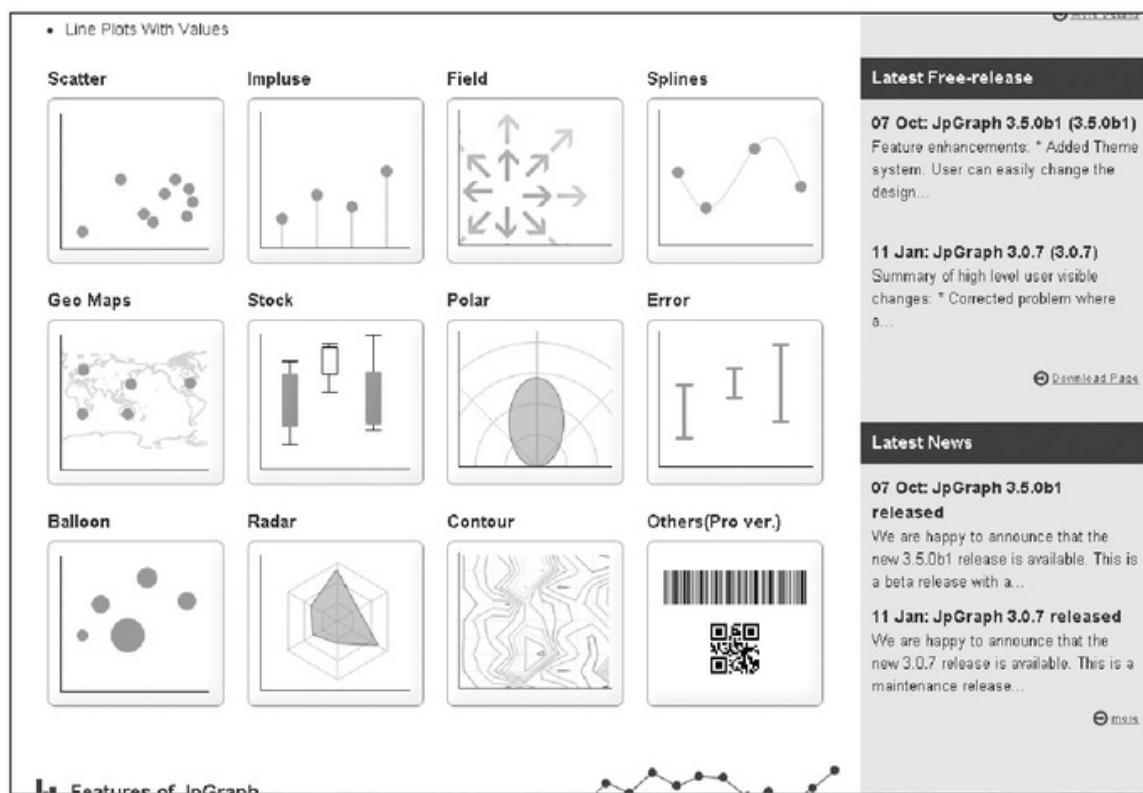


Figura 42. La distribución estándar de JpGraph permite generar numerosos tipos de gráficos.

En cuanto al licenciamiento, JpGraph nos ofrece una versión gratuita y otra paga, que agrega ciertas funcionalidades a la versión básica, entre las que podemos citar:

- Tratamiento de códigos de barra.
- Gráficas tipo velocímetro.
- Gráficas tipo rosa de los vientos.
- Generación de tablas anexas para enumerar los datos ilustrados.

Como podemos observar, las diferencias no son tantas con la edición comercial, y la versión base debería permitirnos llevar a cabo la mayoría de los objetivos incluidos en un desarrollo profesional. Podemos obtener más información acerca de JpGraph, incluyendo una gran cantidad de ejemplos de utilización, en su sitio web oficial: www.aditus.nu/jpgraph.

La imagen del gráfico generado podrá ser **PNG**, **GIF** (siempre y cuando la versión de GD componga este tipo de formato), o **JPG**, de manera predeterminada en ese orden de prioridad. Si necesitamos definir otro orden, modificaremos la constante **DEFAULT_GFORMAT** del archivo **jpg-config.inc.php**:

```
DEFINE("DEFAULT_GFORMAT", "jpg");
```

Este documento contiene múltiples opciones de configuración, aunque los valores de manera predeterminada deberían funcionar para la mayoría de los usuarios.

Los distintos tipos de gráficos mantienen propiedades y métodos comunes entre sí, entre los que podemos citar a modo de ejemplo:

PROPIEDAD	DESCRIPCIÓN
title	Título del gráfico
subtitle	Subtítulo del gráfico
subsubtitle	Segundo subtítulo del gráfico
legend	Leyenda

Tabla 3. Propiedades disponibles de los gráficos generados.

MÉTODO	DESCRIPCIÓN
Add	Se utiliza para añadir un elemento al gráfico.
SetMargin	Define el margen del gráfico.
SetMarginColor	Define el color del margen del gráfico.
SetColor	Define el color del gráfico.
SetBox	Define si el gráfico estará encerrado en una caja.
SetShadow	Permite asignar una sombra al gráfico.
SetGridDepth	Agrega líneas por encima del gráfico o por debajo.
SetAngle	Define el ángulo de inclinación del gráfico.
SetBackgroundImage	Incluye una imagen de fondo al gráfico generado.
SetAxisStyle	Define el estilo de los ejes horizontales y verticales.

Tabla 4. Métodos disponibles de los gráficos generados.

A continuación veremos cómo generar y personalizar algunos de los gráficos disponibles a través de JpGraph.

Gráficos lineales

Quizás es uno de los tipos más utilizados y de los más simples de implementar mediante JpGraph. Lo primero que debemos hacer es incluir los archivos **jpgraph.php** (que sirve de base para la generación de los distintos tipos de gráficos) y **jpgraph_line.php** (específico para los lineales):

```
include 'jpgraph/jpgraph.php';
include 'jpgraph/jpgraph_line.php';
```

Luego, creamos un objeto de tipo **Graph** que actúa como contenedor de los diferentes gráficos. Recibe como argumentos el ancho, alto y el formato del archivo:

```
$grafico = new Graph(450, 250, "auto");
```

Una opción que debemos definir es el tipo de escala por utilizar en los ejes: lineal (**lin**), logarítmica (**log**), textual (**text**), o entera (**int**). La única que no está disponible para ambos ejes es la textual (sólo para X). Mediante el método **SetScale**, indicamos la escala uniendo las constantes; la primera es para el eje X, y la segunda para el Y:

```
$grafico->SetScale("textlin");
```

El método admite cuatro argumentos más, todos ellos opcionales: mínimo y máximo del eje Y, y mínimo y máximo del eje X.

Siguiendo con el ejemplo, generamos un objeto **LinePlot** (que recibe como argumento un array de datos) que será luego agregado al gráfico:

```
$datos = array(10, 12, 3, 14, 21, 13);
$linea = new LinePlot($datos);
$grafico->Add($linea);
```

Por último, enviamos la salida al navegador mediante el método **Stroke**:

```
$grafico->Stroke();
```

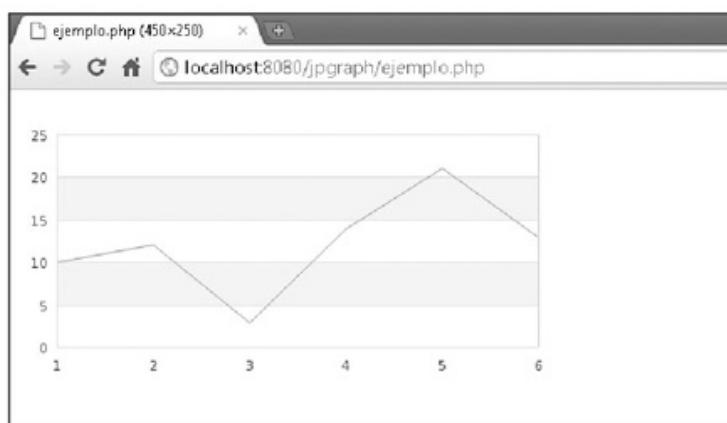


Figura 43. El gráfico generado puede enviarse al navegador del usuario o ser redirigido a un archivo de imagen.

Podemos devolver el gráfico generado a un archivo (válido para todos los gráficos):

```
$grafico->Stroke('imagenes/grafico.png');
```

Además de las ya vistas en el ejemplo anterior, existen muchísimas otras opciones para configurar el aspecto de un gráfico. Por ejemplo, el método **SetMargin** aplicado sobre la propiedad **img** del contenedor nos permite establecer el margen del gráfico (izquierda, derecha, superior e inferior, en ese orden):

```
$grafico->img->SetMargin(45, 10, 10, 45);
```

También podemos utilizar el método **SetShadow** para darle una sombra al contenedor del gráfico. Recibe como argumentos **true** (valor predeterminado, para mostrar el sombreado), ancho (en pixeles) y color (un array que contiene los valores **RGB**):

```
$grafico->SetShadow(true, 2, array(23,12,55));
```

A través de las propiedades **xaxis** e **yaxis**, podemos acceder a los ejes del gráfico y modificar sus características, por ejemplo:

```
$grafico->xaxis->title->Set("titulo eje x");
$grafico->yaxis->title->Set("titulo eje y");
```

Con **SetColor**, definimos el color de la línea (formato hexadecimal) y, con **SetLegend**, el título de referencia a ella:

```
$linea->SetColor("#FF0000");
```

```
$linea->SetLegend("Leyenda aqui");
```

El método **SetFormat** establece un formato para los valores, permitiendo definir la cantidad de decimales o los enteros por visualizar, por ejemplo:

```
$linea->value->SetFormat("%0.2f");
```

Para que esto tenga efecto, debemos indicar que los valores serán exhibidos, lo que es posible lograr a través del método **Show** de la propiedad **value**:

```
$linea->value->Show();
```

El método **setFont** admite tres argumentos: tipo de fuente, estilo y tamaño:

```
$grafico->title->SetFont(FF_VERDANA, FS_BOLD, 10);
```

Entre las fuentes predeterminadas se encuentran:

CONSTANTE	FUENTE
FF_ARIAL	Arial
FF_TIMES	Times Roman
FF_COURIER	Courier new
FF_VERDANA	Verdana
FF_BOOK	Bookman
FF_HANDWRT	Handwriting
FF_COMIC	Sans Comic

Tabla 5. Constantes para definir el tipo de fuente.

Y entre los estilos para modificar el comportamiento por defecto de las fuentes podemos encontrar los siguientes:

CONSTANTE	DESCRIPCIÓN
FS_NORMAL	Normal
FS_BOLD	Negrita
FS_ITALIC	Itálica
FS_BOLDITALIC	Itálica y negrita

Tabla 6. Constante para definir el estilo de la fuente.

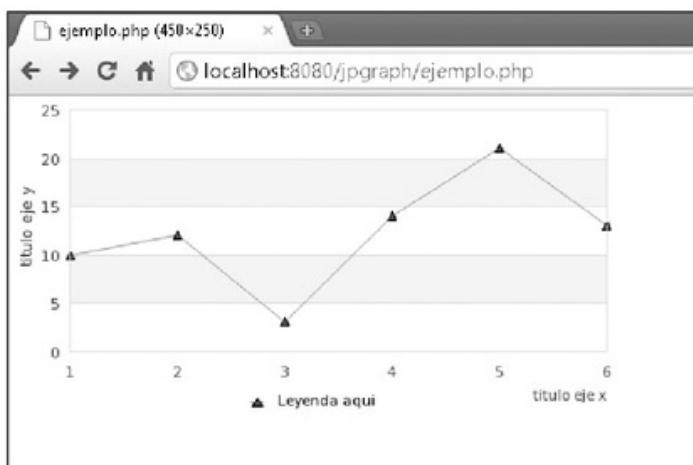
Es posible agregar más constantes para personalizar la fuente incluida en los gráficos. Podemos recabar información acerca de cómo obtener e instalar juegos de caracteres adicionales, en el manual oficial de JpGraph, precisamente en la dirección: <http://jpgraph.net/download/>.

El método **SetType** nos permite personalizar los puntos remarcados en las líneas del gráfico a partir de una serie de opciones. Algunas de las constantes definidas como parámetro de esta función son:

CONSTANTE	DESCRIPCIÓN
MARK_SQUARE	Cuadrado
MARK_UTRIANGLE	Triángulo con la punta hacia arriba
MARK_DTRIANGLE	Triángulo con la punta hacia abajo
MARK_DIAMOND	Rombo
MARK_CIRCLE	Círculo vacío
MARK_FILLED CIRCLE	Círculo relleno
MARK_CROSS	Cruz; signo suma
MARK_STAR	Asterisco
MARK_X	Cruz; signo multiplicación
MARK_LEFTTRIANGLE	Triángulo con la punta hacia la izquierda
MARK_RIGHTTRIANGLE	Triángulo con la punta hacia la derecha
MARK_FLASH	Rayo

Tabla 7. Constante para definir el estilo de la fuente.

```
$linea->mark->SetType(MARK_UTRIANGLE);
```

**Figura 44.** JpGraph nos brinda un gran número de opciones para personalizar el aspecto de los gráficos generados.

Es posible incluir más de una línea en un mismo gráfico:

```
<?php

include 'jpgraph/jpgraph.php';
include 'jpgraph/jpgraph_line.php';

$datosLinea1 = array(2, 4, 5, 7.2, 9, 1);
$datosLinea2 = array(2, 3.2, 4, 5, 2, 1);
```

```

$grafico = new Graph(500, 300, "auto");
$grafico->SetScale("textlin");

$linea1 = new LinePlot($datosLinea1);
$linea1->SetColor("#66CC00");
$linea1->value->Show();
$linea1->value->SetFormat("%0.1f");
$linea1->SetLegend("Linea 1");
$grafico->Add($linea1);

$linea2 = new LinePlot($datosLinea2);
$linea2->SetColor("#FF6666");
$linea2->value->Show();
$linea2->value->SetFormat("%0.1f");
$linea2->SetLegend("Linea 2");
$grafico->Add($linea2);

//leyenda en sentido horizontal
$grafico->legend->SetLayout(LEGEND_HOR);

$grafico->Stroke('imagen.png');

?>

```

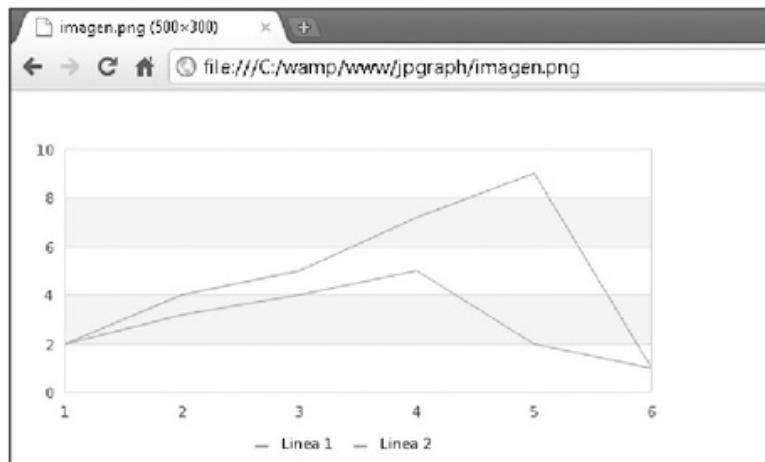


Figura 45. JpGraph admite la inclusión de múltiples tipos de gráficos en una misma imagen.

Los gráficos de líneas permiten representar información de manera sencilla y personalizable, convirtiéndose en una opción válida a la hora de implementar maneras alternativas de presentación de datos en aplicaciones web.

Gráficos de barra

Otro de los tipos de gráficos comúnmente utilizados para representar situaciones en aplicaciones web es el de barras. Para crearlos mediante JpGraph, contamos con la clase **BarPlot**, que recibe como argumento un array de datos, al igual que **LinePlot**:

```
$barra = new BarPlot($datos);
```

Lo primero que debemos hacer antes de instanciar la clase será incluir los archivos correspondientes para trabajar barras (**jpgraph.php** y **jpgraph_bar.php**):

```
include ("jpgraph/jpgraph.php");
include ("jpgraph/jpgraph_bar.php");
```

Luego generamos un contenedor a través de la clase **Graph**:

```
$grafico = new Graph(300, 200, 'auto');
```

Y definimos la escala, al igual que en los ejemplos anteriores con el método **SetScale**:

```
$grafico->SetScale("textlin");
```

Asignamos valores a algunas de las propiedades generales admitidas por la mayoría de los gráficos, en este caso título, leyenda del eje X y leyenda del eje Y:

```
$grafico->title->Set("Titulo del grafico");
$grafico->yaxis->title->Set("Y");
$grafico->xaxis->title->Set("X");
```

Definimos la fuente de datos y la pasamos al constructor de la clase **BarPlot**:

```
$datos = array(24, 12, 11, 15, 3, 14);
$barra = new BarPlot($datos);
```

Los objetos de esta clase tienen sus propios métodos disponibles, entre los que podemos citar los siguientes:

- **SetLegend:** define la leyenda o el título para especificar lo que se está midiendo.

```
$barra->SetLegend("Valores");
```

- **SetFillColor:** nos permite asignar un color a las barras del gráfico.

```
$barra->SetFillColor("#FFCC33");
```

- **SetAbsWidth:** recibe como argumento el número de pixeles de ancho de cada una de las columnas.

```
$barra->SetAbsWidth(30);
```

Opcionalmente, podemos utilizar **SetWidth** que define el ancho en relación al espacio con que cuenta (por ejemplo 90%):

```
$barra->SetWidth("0.9");
```

- El método **Show** aplicado a la propiedad **value** indica que los valores de cada columna serán exhibidos (éste no es el comportamiento predeterminado).

```
$barra->value->Show();
```

- En el mismo sentido, podemos definir el color de la fuente:

```
$barra->value->SetColor('#000000');
```

III REPORTES

El lenguaje PHP no cuenta con un generador de reportes verdaderamente popular y afianzado, sin embargo mantiene una serie de herramientas a disposición de los usuarios, las cuales se encargan de permitir generarlos. Una de estas herramientas puede ser sin dudas **JpGraph**, y otra, **FPDF**, que permite generar documentos en **PDF**.

Definimos la alineación vertical a través del método **SetValuePos**, cuyos posibles valores son **top**, **middle** y **bottom**.

```
$barra->SetValuePos('bottom');
```

Por último, añadimos la barra al contenedor y enviamos el gráfico al navegador para poder visualizarlo:

```
$grafico->Add($barra);
$grafico->Stroke();
```

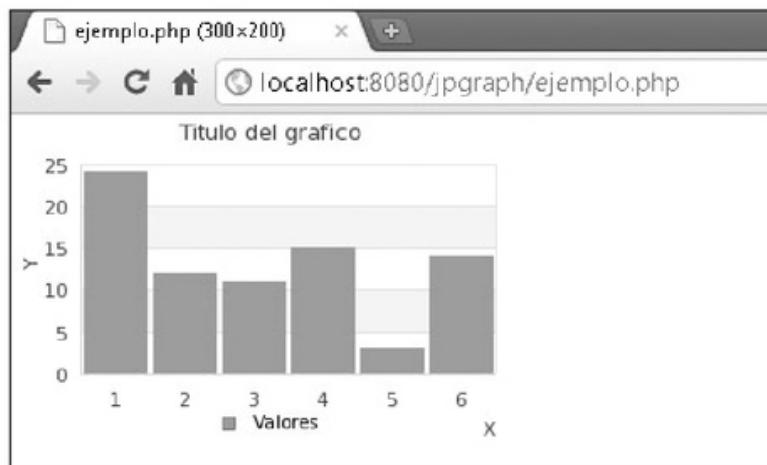


Figura 46. Los gráficos de tipo barra son de los más utilizados en JpGraph a causa de su simpleza y claridad.

En lugar de **SetFillColor**, podemos agregar contraste al color de las barras mediante **SetFillGradient**, que recibe como argumentos los valores hexadecimales de los colores componentes (o un nombre reconocido por JpGraph) y un estilo definido mediante alguna de las constantes disponibles:

- **GRAD_CENTER**
- **GRAD_HOR**
- **GRAD_LEFT_REFLECTION**
- **GRAD_MIDHOR**
- **GRAD_MIDVER**
- **GRAD_RAISED_PANEL**
- **GRAD_RIGHT_REFLECTION**
- **GRAD_VER**
- **GRAD_WIDE_MIDHOR**
- **GRAD_WIDE_MIDVER**

Por ejemplo:

```
$barra->SetFillGradient("#FFFF00", "#993333", GRAD_VER);
```

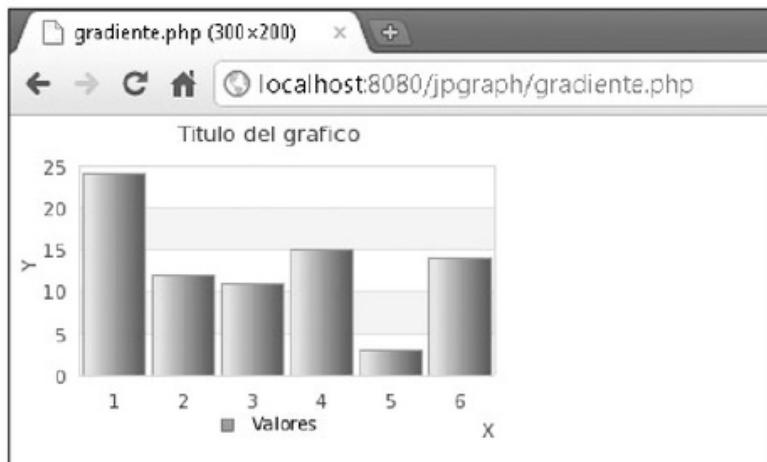


Figura 47. JpGraph admite la combinación de distintos colores para generar el fondo de un gráfico.

Este tipo de gráfico permite variantes, como la inclusión de múltiples barras una sobre otra, o alineadas a través de las clases **GroupBarPlot** y **AccBarPlot**.

Gráficos de torta

Los gráficos de este tipo se generan incluyendo los archivos **jpgraph_pie.php** y **jpgraph_pie3d.php** y, en lugar de la clase **Graph**, se toma como base a **PieGraph**:

index.html

```
<html>
<head>
  <title> graficos con jpgraph </title>
</head>
```

III OPCIONES DISPONIBLES

Debemos saber que la cantidad de alternativas en cuanto a las opciones puestas a disposición del desarrollador para modificar la apariencia o el comportamiento de un gráfico generado con JpGraph es para cualquier usuario, realmente intimidatoria. Esto permite un nivel de detalle muy profundo; algo útil, sin dudas, en ciertas circunstancias.

```
<body>  </body>
</html>
```

grafico.php

```
<?php

include "jpgraph/jpgraph.php";
include "jpgraph/jpgraph_pie.php";
include "jpgraph/jpgraph_pie3d.php";

$datos[] = 10;
$datos[] = 30;
$datos[] = 32;
$datos[] = 15;

$leyendas[] = "2007";
$leyendas[] = "2008";
$leyendas[] = "2009";
$leyendas[] = "2010";

$grafico = new PieGraph(350, 240, "auto");
$grafico->SetMarginColor('#CCCC99');
$grafico->SetShadow();
$grafico->title->Set("Titulo del grafico");

$torta = new PiePlot3D($datos);
$torta->SetSize(0.5);
$torta->SetCenter(0.40);
$torta->SetLabelPos(0.45);
$torta->SetLegends($leyendas);

$torta->value->SetFont(FF_FONT2, FS_NORMAL);
$torta->value->SetColor("#FFFFFF");

$grafico->Add($torta);
$grafico->Stroke();

?>
```

El constructor **PiePlot3D** recibe como argumento un array que deberá contener los datos por graficar (no porcentajes, sino valores).

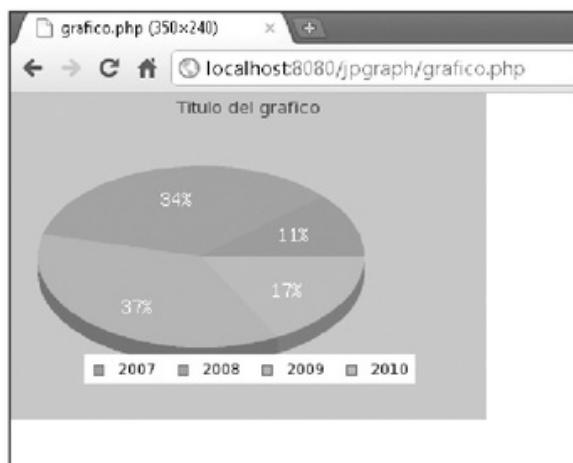


Figura 48. Los gráficos de tipo torta de JpGraph utilizan como clase base a **PieGraph**.

El método **SetSize** puede tomar valores entre 0 y 0.5, y especifica el tamaño de la torta generada en nuestro gráfico:

```
$torta->SetSize(0.5);
```

Por su parte, **SetCenter** define la posición en relación con el contenedor del gráfico (estableceremos 0.5 para ubicarla en el centro):

```
$torta->SetCenter(0.40);
```

El método **SetLabelPos** establece la distancia entre cada título de cada porción y el centro, también es una posición relativa:

```
$torta->SetLabelPos(0.45);
```

Otra opción disponible es **ExplodeAll**, que nos permite separar las porciones de la torta agregando un margen intermedio:

```
$torta->ExplodeAll();
```

Al generar el gráfico, los valores pasados como argumento a **PiePlot3D** se convierten en porcentajes de manera automática.

Anillos

Estos gráficos son una variación de los anteriores y se implementan de manera similar. En primer lugar, debemos utilizar la clase **PiePlotC** en lugar de **PiePlot3D**:

```
<?php

include "jpgraph/jpgraph.php";
include "jpgraph/jpgraph_pie.php";
include "jpgraph/jpgraph_pie3d.php";

$datos[] = 10;
$datos[] = 30;
$datos[] = 32;
$datos[] = 15;

$leyendas[] = "2007";
$leyendas[] = "2008";
$leyendas[] = "2009";
$leyendas[] = "2010";

$grafico = new PieGraph(350, 240, "auto");
$grafico->SetMarginColor('#CCCC99');
$grafico->SetShadow();
$grafico->title->Set("Titulo del grafico");

$anillo = new PiePlotC($datos);
$anillo->SetSize(0.4);
$anillo->SetCenter(0.40);
$anillo->SetLabelPos(0.75);
$anillo->SetLegends($leyendas);
```



MÉRITOS PROPIOS

Es importante saber que la herramienta llamada JpGraph es un claro ejemplo de una aplicación que utiliza librerías externas para funcionar (GD), pero que, sin embargo, se ha sabido ganar un nombre propio y un prestigio en relación con los desarrolladores de aplicaciones que necesitan incorporar gráficos estadísticos en sus programas.

```
$anillo->value->SetFont(FF_FONT2, FS_NORMAL);
$anillo->value->SetColor("#FFFFFF");
$grafico->Add($anillo);
$grafico->Stroke();

?>
```

Un método extra de utilidad es **SetMidColor**, que permite definir el color de fondo del círculo central del anillo, muy útil en algunas ocasiones:

```
$anillo->SetMidColor('#CCCC99');
```

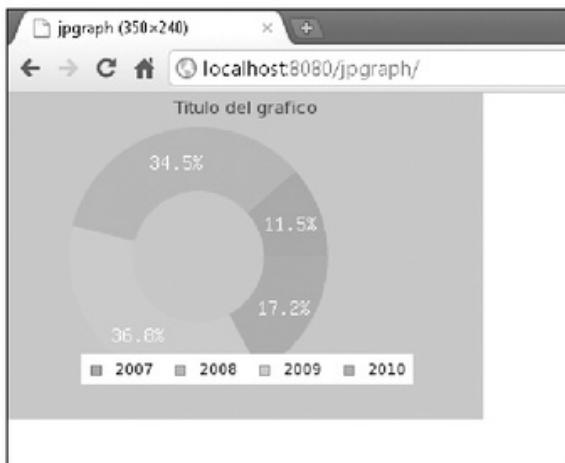


Figura 49. Los gráficos de tipo anillo poseen una estructura similar a los de tipo torta y, en JpGraph, generalmente, tienen las mismas características.

Si este valor es igual al establecido en **SetMarginColor**, el fondo del gráfico quedará establecido con un color uniforme.

Leds

Un led podría definirse en este contexto como una serie de puntos iluminados que contrastan con los demás, formando caracteres. Fueron incluidos recientemente en las últimas versiones de la librería, y los signos soportados para incluir en esta clase de gráficos son:

- Numéricos (del 0 al 9)
- Puntos (.)
- Espacios ()
- Numerales (#)
- Letras mayúsculas (de la A a la Z)

Inicialmente, debemos incluir el archivo **jpgraph_led.php** y generar un objeto **DigitalLED74** (que recibe como argumentos opcionales el radio, de manera predeterminada 2, y el margen, de manera predeterminada 0.6):

```
$led = new DigitalLED74();
```

El método **StrokeNumber** en este caso recibe como argumento el texto por generar:

```
$led->StrokeNumber(rand(1000000000, 900000000));
```



Figura 50. El soporte para Leds se incluyó recién en las últimas versiones de JpGraph y está en pleno proceso de desarrollo.

Además, podemos incluir como segundo argumento una constante que representa el color del texto generado:

- LEDC_RED** (Rojo)
- LEDC_GREEN** (Verde)
- LEDC_BLUE** (Azul)
- LEDC_YELLOW** (Amarillo)
- LEDC_GRAY** (Gris)

```
$led->StrokeNumber('DIA #1', LEDC_GREEN);
```

CAPTCHA

Vimos en capítulos anteriores cómo generar imágenes CAPTCHA desde PHP y cómo incluirlas en el contexto de un formulario para ingreso y validación de usuarios. JpGraph aporta una implementación sencilla y potente para, en pocas líneas de código, generar aplicaciones funcionales. El archivo que debemos incluir en este caso es **jpgraph_antispam.php**:

```
include "jpgraph/jpgraph_antispam.php";
```

Y luego inicializar un objeto **AntiSpam**:

```
$spam = new AntiSpam();
```

La cadena de caracteres contenida en la imagen puede ser definida de dos maneras: al azar (mediante el método **Rand**, que recibe como argumento la longitud del texto) o de forma predeterminada (a través del método **Set**, que recibe como argumento el string):

```
$spam->Rand($longitud);
```

```
$spam->Set($cadena);
```

En cualquier caso y para comparar lo que ingresa el usuario con lo que dice la imagen, será necesario guardar el contenido (en una variable de sesión, por ejemplo):

```
session_start();
$_SESSION['cadena'] = $spam->Rand($longitud);
```

```
session_start();
$_SESSION['cadena'] = $spam->Set($cadena);
```

Por último, imprimimos la imagen generada:

```
$spam->Stroke();
```

Suponiendo que lo anterior se almacenara dentro de un archivo llamado **grafico.php**, en el formulario de ingreso compararíamos el texto enviado por el usuario con lo almacenado en la variable **cadena**:

```
<?php
session_start();
```

```

if (count($_POST)) {
    if ($_SESSION['cadena'] == $_POST['texto']) {
        die ('<li>Ingreso correcto');
    } else {
        echo '<li>Error en ingreso';
    }
}

echo '<br />';

echo '<form method="post">';
echo '<input type="text" name="texto" id="texto">';
echo '<input type="submit">';
echo '</form>';

?>

```



Figura 51. El soporte para generar imágenes CAPTCHA propuesto por JpGraph es sencillo de implementar y a la vez muy potente.

En el **Capítulo 8**, dedicado a PEAR, se muestran otras variantes similares para generar este tipo de soluciones de gráficos dinámicos.

Mapas HTML

Un mapa HTML sirve para ligar partes de una imagen a diferentes acciones (enlaces a distintas páginas o archivos, por ejemplo). En JpGraph, es posible crear este tipo de elementos que se diferencian del resto de los gráficos, principalmente, porque se genera no sólo el archivo correspondiente a la imagen, sino un fragmento HTML que deberá ser incluido. Es posible incluir más de un mapa en una misma página. Suponiendo que lo generemos desde **mapa.php**, se invocaría de la siguiente manera:

index.php

```
<html>
<head>
<title> jpgraph - mapa </title>
</head>
<body> <?php include 'mapa.php'; ?> </body>
</html>
```

El contenido de **mapa.php** no difiere, en cuanto a lo que sería habitual, en lo relativo a la creación de gráficos: las únicas diferencias son la utilización de los métodos **SetCSIMTargets** y **StrokeCSIM**. Cada elemento del mapa necesita dos valores (un enlace válido y un título) que se asignan mediante el método **SetCSIMTargets**. En lugar de **Stroke**, utilizamos el método **StrokeCSIM**, que debe recibir el nombre del archivo actual:

```
<?php

include ("jpgraph/jpgraph.php");
include ("jpgraph/jpgraph_bar.php");

$grafico = new Graph(500, 200, 'auto');

for ($c=0;$c<12;$c++) {
    $enlaces[$c] = '/detalle.php?mes=' . ($c+1);
    $titulos[$c] = 'Mes numero ' . ($c+1);
    $datos[$c] = rand(0, 100);
}

$grafico->SetScale("textlin");

$grafico->title->Set("Titulo del grafico");
$grafico->yaxis->title->Set("Y");
$grafico->xaxis->title->Set("X");

$barra = new BarPlot($datos);

$barra->SetLegend("Valores");

$barra->SetFillGradient('yellow', 'white', GRAD_VER);
```

```

$barra->SetWidth("0.9");

$barra->value->Show();

$barra->value->SetFormat("%0.0f");

$barra->value->SetColor('#000000');

$barra->SetValuePos('bottom');

$barra->SetCSIMTargets($enlaces, $titulos);

$grafico->Add($barra);

$grafico->StrokeCSIM('mapa.php');

?>

```

El código fuente de **index.html** contiene el mapa HTML incrustado en el contenido:



The screenshot shows a browser window titled "jpgraph - mapa" with the URL "view-source:localhost:8080/jpgraph/". The page content is a block of HTML code representing a map with 8 numbered regions. Each region is a polygon defined by coordinates and a href attribute linking to "/detalle.php?mes=x" where x is the region number. The regions are labeled "Mes numero 1" through "Mes numero 8".

```

1 <html>
2   <head>
3     <title> jpgraph - mapa </title>
4   </head>
5   <body> <map name="_mapname61670_" id="_mapname61670_">
6     <area shape="poly" coords="42, 150, 42, 72, 75, 72, 75, 150" href="/detalle.php?mes=1" title="Mes numero 1" alt="Mes numero 1" />
7     <area shape="poly" coords="79, 150, 79, 142, 112, 142, 112, 150" href="/detalle.php?mes=2" title="Mes numero 2" alt="Mes numero 2" />
8     <area shape="poly" coords="115, 150, 115, 55, 148, 55, 148, 150" href="/detalle.php?mes=3" title="Mes numero 3" alt="Mes numero 3" />
9     <area shape="poly" coords="152, 150, 152, 123, 185, 123, 185, 150" href="/detalle.php?mes=4" title="Mes numero 4" alt="Mes numero 4" />
10    <area shape="poly" coords="189, 150, 189, 33, 222, 33, 222, 150" href="/detalle.php?mes=5" title="Mes numero 5" alt="Mes numero 5" />
11    <area shape="poly" coords="225, 150, 225, 137, 258, 137, 258, 150" href="/detalle.php?mes=6" title="Mes numero 6" alt="Mes numero 6" />
12    <area shape="poly" coords="262, 150, 262, 129, 295, 129, 295, 150" href="/detalle.php?mes=7" title="Mes numero 7" alt="Mes numero 7" />
13    <area shape="poly" coords="299, 150, 299, 77, 332, 77, 332, 150" href="/detalle.php?mes=8" title="Mes numero 8" alt="Mes numero 8" />

```

Figura 52. JpGraph genera fragmentos HTML para implementar mapas.

III ORIGEN DE LOS DATOS

Los datos graficados a partir de la utilización de la librería JpGraph pueden ser de origen variado. La única condición que requieren es poder representarse a través de arrays (matrices) PHP, algo que por supuesto de ningún modo es un inconveniente y que podemos generar dentro de la aplicación antes de invocar a JpGraph.

El gráfico se genera utilizando valores aleatorios según lo establecimos en el código, por lo cual la salida será diferente en cada ejecución.



Figura 53. Los mapas HTML son aplicables a distintos tipos de gráficos.

Utilización de gráficos en caché

En ocasiones, los gráficos generados por un script no varían de forma dinámica, por lo que invocar su creación a través de las clases y métodos disponibles a partir de JpGraph consume recursos innecesariamente. Para administrar el uso de la caché de gráficos, existen las constantes **USE_CACHE** (habilitar la memoria caché), **READ_CACHE** (utilizar los gráficos que están en caché) y **CACHE_DIR** (ruta absoluta a un directorio temporal, que debe contar con permisos de escritura), disponibles todas en el archivo **jpg-config.php**:

```
DEFINE("USE_CACHE", true);
DEFINE("READ_CACHE", true);
DEFINE("CACHE_DIR", "/imagenes/cache/");
```

Si la caché está habilitada, podremos utilizarla incluyendo dos argumentos en el constructor del objeto correspondiente al contenedor:

```
$grafico = new Graph($ancho, $alto, $nombreImagenCache, $tiempoExpiracion);
```

El tercer argumento es el nombre de la imagen en el directorio temporal (por ejemplo, **grafico.png**), y el cuarto es el tiempo en minutos antes de actualizar la imagen (0 para no actualizar nunca). Si la variable **nombreImagenCache** es igual a **auto**, se utilizará como nombre de imagen el nombre del script más la extensión que corresponda. Por

ejemplo, si el gráfico se genera desde la página **grafico.php**, el archivo almacenado se llamará **grafico.png**, si es que utilizamos **PNG** como formato. Si la imagen no se encuentra disponible en caché, se ejecuta el script correspondiente para generarla.

En el caso de los mapas, deberemos invocar al método **CheckCSIMCache** que recibe como argumentos **nombreImagenCache** y **tiempoExpiracion**:

```
$grafico = new Graph($ancho, $alto);
$grafico->CheckCSIMCache($nombreImagenCache, $tiempoExpiracion);

//codigo

$grafico->StrokeCSIM();
```

En cuanto a las constantes, contamos con **CSIMCACHE_DIR**, similar a **CACHE_DIR**, pero exclusivo para mapas:

```
DEFINE("CSIMCACHE_DIR", "/imagenes/cache/mapas/");
```

Recordemos siempre que la ruta deberá ser absoluta, y que el directorio tendrá que contar con permisos de lectura y escritura.

Ejemplo práctico

En el siguiente ejemplo implementamos un contador de visitas. Utilizaremos la siguiente base de datos:

```
create database estadisticas;
use estadisticas;

create table visitas (
    id int(11) primary key not null auto_increment,
    fecha date,
    pagina varchar(100),
    qs varchar(100),
    total_visitadas int(11) default 1
);
```

En cada página incluiremos la siguiente línea:

```
<?php include 'contador.php'; ?>
```

contador.php

```
<?php

include 'conexion.php';

$paginaActual = mysql_real_escape_string($_SERVER['PHP_SELF']);
$qsActual = mysql_real_escape_string($_SERVER['QUERY_STRING']);
$fechaActual = date("Y-m-d");

$res = mysql_query("select * from visitas where fecha = '$fechaActual' and
    pagina = '$paginaActual' and qs = '$qsActual'");
if (mysql_num_rows($res)) {
    $res = mysql_query("update visitas set total_visitadas = total_visitadas +1
        where fecha = '$fechaActual' and pagina = '$paginaActual' and qs =
        '$qsActual'");
} else {
    $res = mysql_query("insert into visitas (fecha, pagina, qs) values
        ('$fechaActual', '$paginaActual', '$qsActual')");
}

?>
```

conexion.php

```
<?php
```

TIPOS DE ARCHIVOS

La librería GD soporta múltiples formatos para la generación y tratamiento de imágenes, entre ellos **GIF**. Por un tema comercial (el algoritmo **LZW** utilizado por este formato es propiedad de la empresa **Unisys**), las últimas versiones no dan soporte, algo que eventualmente puede hacer que tengamos que modificar nuestros programas.

```

$hostname = "localhost";
$database = "estadisticas";
$username = "root";
$password = "";

$cxn = mysql_connect($hostname, $username, $password) or die('No encuentro
    servidor MySql');
$cxn = mysql_select_db($database) or die('No encuentro base de datos');

?>

```

En el campo **qs** de la tabla, se almacena el **querystring** mediante el cual se accedió a la página. Esto puede ser útil en determinadas situaciones en las que se muestra un contenido u otro, dependiendo del valor pasado por URL. Por ejemplo:

```

productos.php?idProducto=1
productos.php?idProducto=2
productos.php?idProducto=N

```

Para visualizar el gráfico correspondiente a una fecha determinada, accedemos a la página **grafico.php**, que contendrá un buscador para seleccionar el mes y el año por graficar. Primero generamos las dos listas:

```

$mes[1] = 'Enero';
$mes[] = 'Febrero';
$mes[] = 'Marzo';
$mes[] = 'Abril';
$mes[] = 'Mayo';
$mes[] = 'Junio';
$mes[] = 'Julio';
$mes[] = 'Agosto';
$mes[] = 'Septiembre';
$mes[] = 'Octubre';
$mes[] = 'Noviembre';
$mes[] = 'Diciembre';

$selectM = "<select name='mes' id='mes'>";
foreach ($mes as $c => $v) {

```

```

$selected = $_POST['mes'] == $c ? 'selected' : '';
$selectM .= "<option $selected value='$c'$v </option>";
}
$selectM .= "</select>";

$selectA = "<select name='anio' id='anio'>";
for ($c=date("Y");$c>(date("Y")-5);$c-) {
    $selected = $_POST['anio'] == $c ? 'selected' : '';
    $selectA .= "<option $selected value='$c'$c </option>";
}
$selectA .= "</select>";

```

Y luego imprimimos el formulario:

```

echo "<form method='post'>";
echo $selectM;
echo $selectA;
echo "<input type='submit' value='Ver visitas'>";
echo "</form>";

```

Si se ejecuta una búsqueda, se carga una imagen desde el archivo **generarImagen.php**, que recibe como argumentos el mes y el año seleccionados:

```

if (count($_POST)) {
    echo "<img src='generarImagen.php?mes=$_POST[mes]&anio=$_POST[anio]'>";
}

```

Ya en esta página, incluimos el archivo de conexión junto con los necesarios para elaborar el gráfico:

```

include ("conexion.php");
include ("jpgraph/jpgraph.php");
include ("jpgraph/jpgraph_bar.php");

```

Los datos por graficar se recuperan desde la base de datos y se almacenan en el array **pagina**, cuyas claves serán los nombres de las páginas, y los valores, el número de visitas, de la siguiente manera:

```

$res = mysql_query("select * from visitas where DATE_FORMAT(fecha, '%c-%Y') 
= '$_GET[mes]-$_GET[anio]'");
if (mysql_num_rows($res)) {
    while ($row = mysql_fetch_array($res)) {
        $pagina[$row['pagina']] += $row['total_visitas'];
    }
}

arsort($pagina);
} else {
    $pagina[' No hay visitas disponibles'] = '';
}

```

En este caso no tenemos en cuenta el campo **qs**: sólo se calculan las visitas por página.



Figura 54. Podemos incluir JpGraph en distintos tipos de proyectos sin alterar su funcionamiento.

Ahora comenzamos a delinejar el gráfico: primero definimos sus dimensiones (podemos observar que la altura está relacionada con la cantidad de páginas, ya veremos la razón) y su escala:

```

$alto = count($pagina)*50;
$grafico = new Graph(600, $alto, 'auto');
$grafico->SetScale("textlin");

```

Luego los márgenes:

```

$margenSuperior = 10;
$margenInferior = 20;
$margenIzquierdo = 10;
$margenDerecho = 15;

```

```
$grafico->Set90AndMargin($margenIzquierdo, $margenDerecho, $margenSuperior,
    $margenInferior);
```

El método **Set90AndMargin** es de mucha importancia en este ejemplo: su inclusión permite definir el margen del gráfico y, además, rotarlo 90 grados, lo que implica que las barras aparecerán en sentido horizontal y no vertical. Por esta razón, en lo que resta del código cuando se referencia al eje X, en realidad, se trabaja sobre el Y, y viceversa. Continuando con la generación del gráfico, le asignamos una sombra mediante **SetShadow**:

```
$grafico->SetShadow();
```

En el eje X, estarán los nombres de las páginas, o sea, las claves del array **pagina**:

```
$datosX = array_keys($pagina);
$grafico->xaxis->SetTickLabels($datosX);
$grafico->xaxis->SetFont(FF_VARDANA, FS_NORMAL, 7);
$grafico->xaxis->SetLabelAlign('left', 'center');
$grafico->xaxis->SetLabelMargin(0);
```

El método **SetTickLabels** nos permite incluir títulos personalizados para los ítems del eje, y **SetGrace** indica el margen inferior de las barras:

```
$grafico->yaxis->scale->SetGrace(10);
```

Ahora ocultamos los títulos del eje Y a través del método **Hide**:

```
$grafico->yaxis->Hide();
```

Por último, creamos y agregamos la barra al gráfico:

```
$datosY = array_values($pagina);
$barra = new BarPlot($datosY);
$barra->SetFillColor("#FFFFCC");
$barra->value->Show();
```

```
$barra->value->SetFont(FF_VARDANA, FS_NORMAL, 7);
$barra->value->SetColor("#339933", "darkred");
$barra->value->SetFormat('%.0f visitas');
$barra->SetWidth("0.9");

$grafico->Add($barra);
```

E imprimimos el resultado:

```
$grafico->Stroke();
```



Figura 55. La generación dinámica de gráficos hace de JpGraph una alternativa muy útil a la hora de representar información tomada de una base de datos.

Las posibilidades brindadas por JpGraph son casi infinitas, y, en cada nueva versión, se agregan más y más características que perfeccionan lo que ya se ofrecía o dan nuevas alternativas a la hora de generar gráficos. Se trata, sin dudas, de una opción muy recomendable y de nivel profesional.

PEAR

PEAR provee varios paquetes para manipular imágenes, entre los que citamos:

- **Image_3D:** generación de objetos tridimensionales.
- **Image_Barcod:** generación de códigos de barra.
- **Image_Canvas:** generación de imágenes.
- **Image_Color:** administración y manejo de colores.

- **Image_Color2:** **Image_Color** para PHP5.
- **Image_GIS:** visualización basada en datos geográficos.
- **Image_Graph:** generación de gráficos para PHP4 y PHP5.
- **Image_MonoBMP:** manipulación de imágenes **BMP** monocromáticas.
- **Image_Puzzle:** genera piezas de rompecabezas a partir de una imagen.
- **Image_Remote:** recupera información acerca de imágenes en servidores remotos.
- **Image_Text:** inclusión de texto en imágenes.
- **Image_Tools:** herramientas para manipulación de imágenes.
- **Image_Transform:** manipulación y conversión de imágenes con distintas librerías.
- **Image_WBMP:** manipulación de imágenes tipo **WBMP**.
- **Image_XBM:** manipulación de imágenes tipo **XBM**.
- **XML_image2svg:** convierte una imagen a **SVG**.
- **XML_svg2image:** genera una imagen a partir de un archivo **SVG**.

Una de las opciones más utilizadas es **Image_Graph** (también conocida como **GraPHPite**), que nos permite generar gráficos estadísticos de distinto tipo. Para comenzar a utilizarla, debemos instalar los paquetes **Image_Canvas** e **Image_Color** (este último suele instalarse de manera automática junto con el primero):

```
C:\> pear install Image_Canvas
C:\> pear install Image_Graph
```

The screenshot shows the Pear package repository interface. At the top, there's a navigation bar with links for Main, Support, Documentation, Packages, Package Proposals, Developers, Bugs, Register, and Login. Below the navigation bar, there's a search bar with the placeholder "Search for" and a dropdown menu set to "in the Packages". The main content area has a header "Top Level :: Images" and "Package Information: Image_Graph". Below this, there are tabs for Main, Download, Documentation, Bugs, and Trackbacks, with "Main" being the active tab. A note says "This package is not maintained, if you would like to take over please go to [this page](#)". There are two main sections: "» Summary" and "» Current Release". The "Summary" section contains a brief description: "A package for displaying (numerical) data as a graph/chart/plot.". The "Current Release" section lists "0.8.0 (alpha)" released on 2010-10-05, with a link to the Changelog. To the right of these sections are "» License" (set to LGPL) and "» Bug Summary". The "Bug Summary" section provides statistics: "Package Maintenance Rank: 175 of 203 packages with open bugs", "Number of open bugs: 23 (84 total bugs)", "Average age of open bugs: 1079 days", "Oldest open bug: 1859 days", and "Number of open feature requests: 7 (16 total feature requests)". Below these sections is a link to "Report a new bug to Image_Graph". Further down, there's a "» Description" section with a detailed description of the package: "Image_Graph provides a set of classes that creates graphs/plots/charts based on (numerical) data. Many different plot types are supported: Bar, line, area, step, impulse, scatter, radar, pie, map, candlestick, band, box & whisker and smoothed line, area and radar plots. The graph is highly customizable, making it possible to get the exact look and feel that is required. The output is controlled by a Image_Canvas, which facilitates easy output to many different output formats, amongst others, GD (PNG, JPEG, ...)." The entire page is framed by a light gray border.

Figura 56. El repositorio de clases oficial de PHP nos provee numerosas opciones para el tratamiento de archivos de imagen.

Podemos obtener más información acerca de cómo instalar paquetes PEAR en el **Capítulo 8**. Una vez completado el proceso, para generar gráficos, debemos incluir el archivo **Graph.php**:

```
include 'Image/Graph.php';
```

Para instanciar un objeto de tipo **Image_Graph**, podemos utilizar el patrón de diseño **Factory**, cuyo segundo argumento en este caso es un array que define las medidas del gráfico (ancho y alto):

```
$grafico =& Image_Graph::factory('graph', array(600, 300));
```

Podemos ver un objeto **Image_Graph** como un marco en el cual luego ubicaremos los distintos tipos de gráficos.

El origen de los datos se establece a partir de un objeto de tipo **dataset**, seteado de la siguiente manera:

```
$datos =& Image_Graph::factory('dataset');
```

Las coordenadas de los puntos por graficar se pueden incluir mediante el método **addPoint**, como vemos en estas tres líneas:

```
$datos->addPoint($titulo1, $valor1);
$datos->addPoint($titulo2, $valor2);
$datos->addPoint($tituloN, $valorN);
```

Luego, establecemos el tipo de gráfico (en este caso uno de barras) y lo vinculamos al set de datos:



PAQUETES PARA IMÁGENES EN PEAR

Es importante recordar que si queremos investigar más sobre los paquetes para imágenes en PEAR, podemos obtener más información acerca de las características y objetivos de cada uno de los paquetes dedicados al tratamiento y a la generación de imágenes en el sitio de PEAR, en la dirección <http://pear.php.net/packages.php?catid=12&catname=Images>.

```
$espacio =& $grafico->addNew('plotarea');
$barra =& $espacio->addNew('bar', &$datos);
```

La variable **espacio** contiene una referencia al lugar en donde será incluido el gráfico de barras. Podemos utilizar **addNew** repetidas veces para incluir más de un tipo de gráfico. Enviamos la imagen generada al navegador, a través del método **done**:

```
$grafico->done();
```

Para enviar la salida a un archivo:

```
$grafico->done(array('filename' => 'imagen.png'));
```

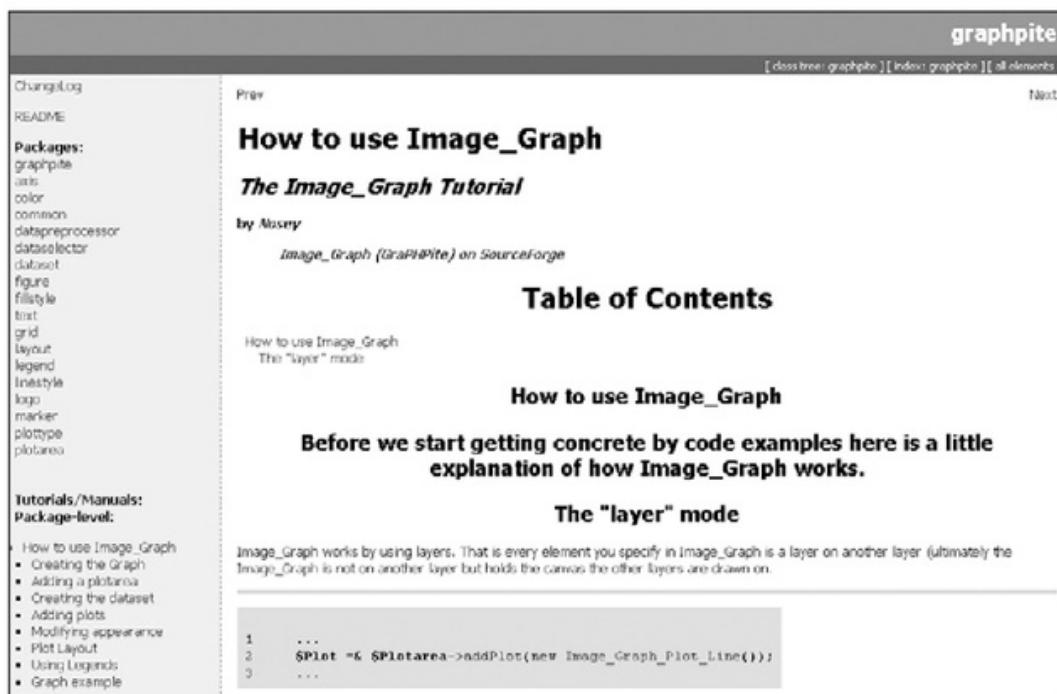


Figura 57. La generación de imágenes con *Image_Graph* nos permite implementar soluciones de manera rápida y consistente.

Además de **bar** (o su alias **Image_Graph_Plot_Bar**), existen las siguientes constantes para definir el tipo de gráfico:

- **line** o **Image_Graph_Plot_Line**
- **area** o **Image_Graph_Plot_Area**
- **smooth_line** o **Image_Graph_Plot_Smoothed_Line**
- **smooth_area** o **Image_Graph_Plot_Smoothed_Area**

- pie o `Image_Graph_Plot_Pie`
- step o `Image_Graph_Plot_Step`
- impulse o `Image_Graph_Plot_Impulse`
- dot o scatter o `Image_Graph_Plot_Dot`
- radar o `Image_Graph_Plot_Radar`
- `Image_Graph_Plot_CandleStick`
- `Image_Graph_Plot_Band`

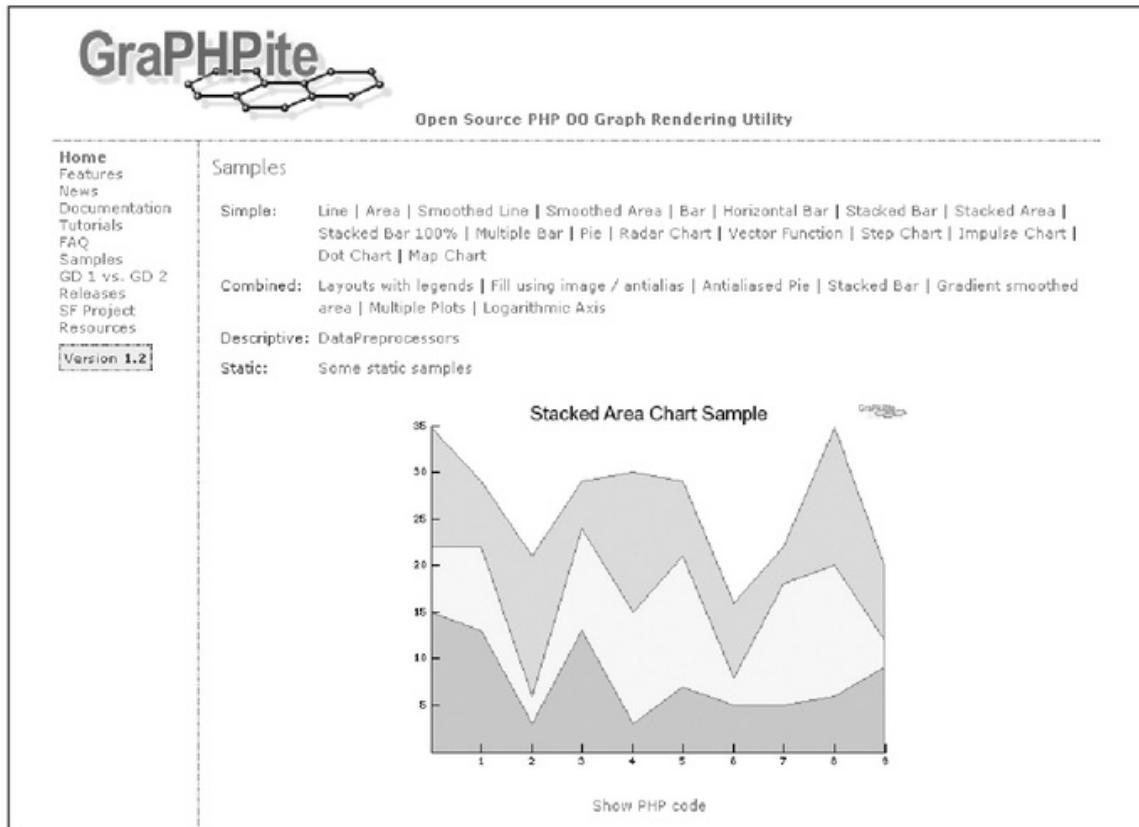


Figura 58. *Image_Graph* provee un buen número de gráficos prefijados, listos para incluir en nuestros proyectos.

Otras opciones disponibles para modificar la apariencia de los gráficos son:

- Definir un título para el gráfico:

```
$barra->setTitle($titulo);
```

- Especificar el tipo y el tamaño de la fuente:

```
$fuente =& $grafico->addNew('ttf_font', 'Verdana');
$fuente->setSize(8);
```

```
$grafico->setFont($fuente);
```

- Con **setLineColor**, definimos el color del borde de las barras:

```
$barra->setLineColor('#FFFF00');
```

- Con **setFillColor**, establecemos el color de fondo de todas las barras:

```
$barra->setFillColor('#FF6633');
```

Para indicar un color distinto para cada una, disponemos de la clase **Image_Graph_Fill_Array**. Cada llamada a **addColor** define el color para la siguiente:

```
$colores = & Image_Graph::factory('Image_Graph_Fill_Array');
$colores->addColor('#FF6600');
$colores->addColor('#FF3300');
$colores->addColor('#FF0000');

$barra->setFillStyle($colores);
```

Otra opción es la de difuminar dos colores (gradiente):

```
$gradiente = & Image_Graph::factory('gradient', array($constante, $color1,
    $color2));
$barra->setFillStyle($gradiente);
```

En donde la variable **constante** indica el estilo por utilizar y puede ser una de las siguientes opciones:

IMAGE_GRAPH_GRAD_HORIZONTAL
IMAGE_GRAPH_GRAD_VERTICAL
IMAGE_GRAPH_GRAD_HORIZONTAL_MIRRORED
IMAGE_GRAPH_GRAD_VERTICAL_MIRRORED
IMAGE_GRAPH_GRAD_DIAGONALLY_TL_BR
IMAGE_GRAPH_GRAD_DIAGONALLY_BL_TR
IMAGE_GRAPH_GRAD_RADIAL

```
$gradiente =& Image_Graph::factory('gradient',
array(IMAGE_GRAPH_GRAD_VERTICAL, '#669900', '#666600'));
```

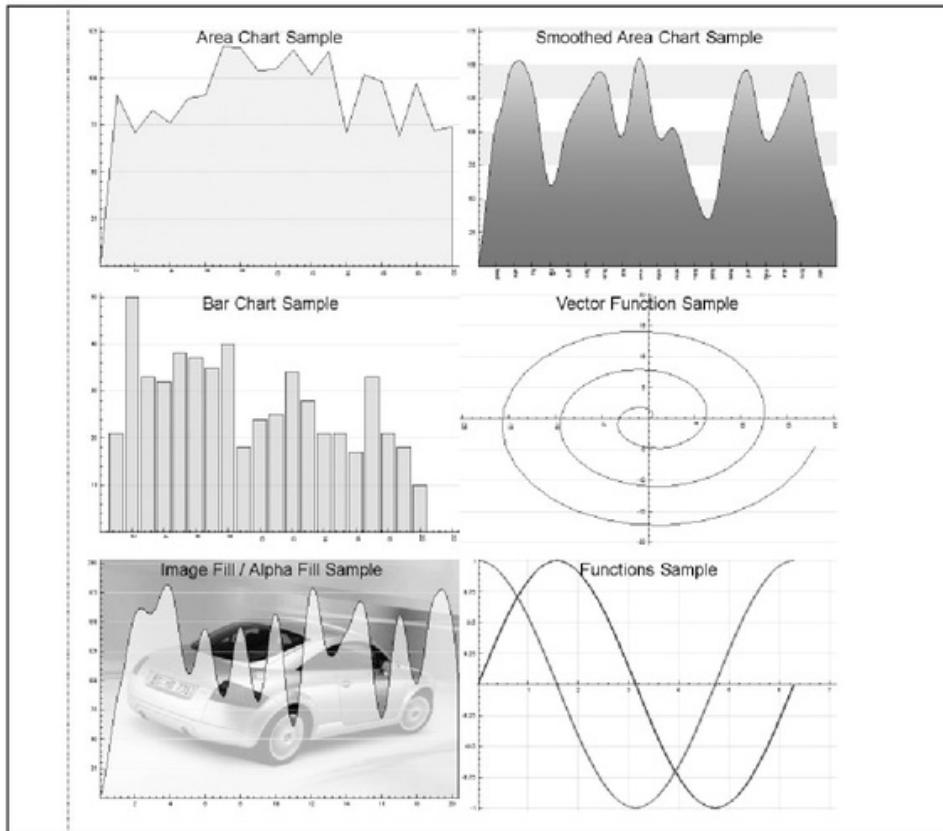


Figura 59. Image_Graph se caracteriza por su simplicidad y, en cada nueva versión, incorpora más y más opciones sin complejizar el proceso de desarrollo.

En cualquier caso, podemos indicar un nivel de transparencia para los colores utilizando la sintaxis **color@nivel**:

```
$barra->setFillColor('#FF6633@0.1');
```

```
$fill->addColor('#FF3300@0.4');
```

El nivel indica el grado (porcentaje) de transparencia y varía entre 0 y 1 (mientras más se acerca a 1, menos transparente es el color).

- Con **setBackgroundColor**, indicamos el color de fondo del gráfico:

```
$barra->setBackgroundColor('#330099');
```

- Otra opción es ubicar una imagen como fondo, a través de la clase **Image_Graph_Fill_Image**. En este caso, la imagen se ubicará detrás de nuestro gráfico, y éste se le superpondrá. Recordemos que la imagen no debe confundirse con el gráfico en sí, y será lo más liviana posible.

```
$fondo =& Image_Graph::factory('Image_Graph_Fill_Image', 'image.jpg');
$espacio->setFillStyle($fondo);
```

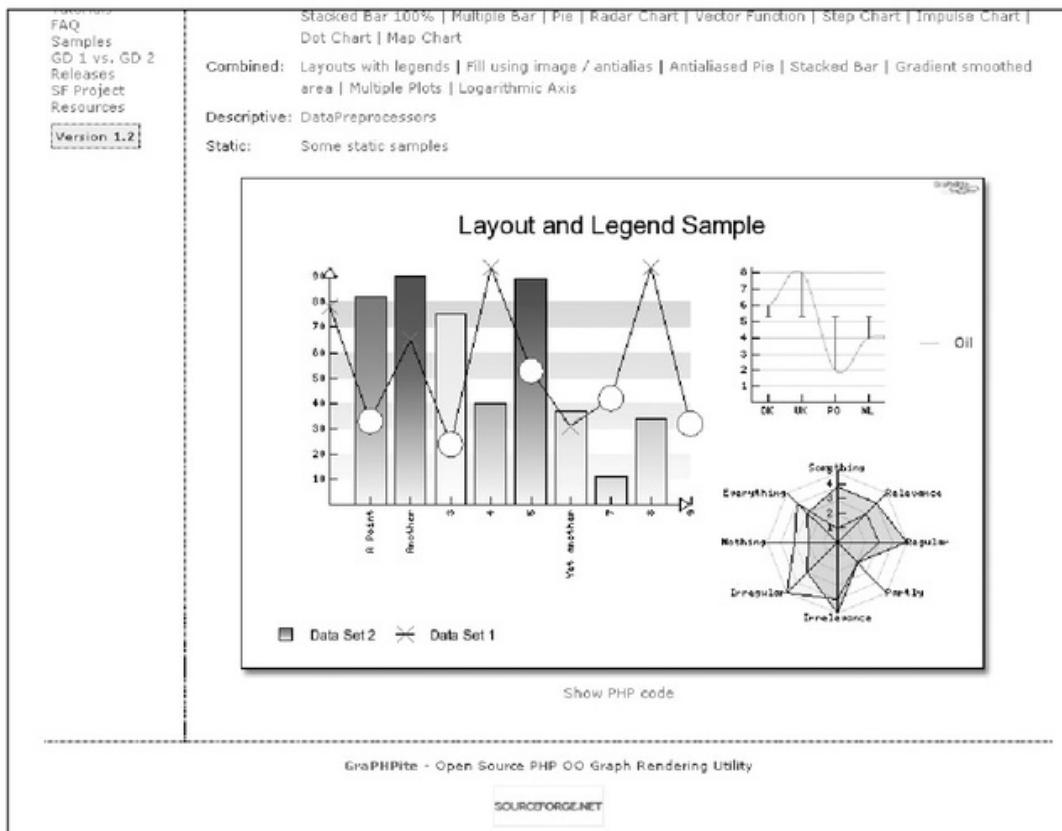


Figura 60. La mayoría de los tipos de gráficos disponibles poseen, generalmente, los mismos métodos y propiedades.

- Para acceder a los ejes del gráfico, se utilizan las constantes **IMAGE_GRAPH_AXIS_X** o **IMAGE_GRAPH_AXIS_Y** de la siguiente manera:

```
$x =& $Plotarea->getAxis(IMAGE_GRAPH_AXIS_X);
$x->setTitle($titulo);
```

Éstas son sólo algunas de las múltiples alternativas de la librería Image_Graph que, sin dudas, es una opción comparable a JpGraph tanto en potencia como en versatilidad. Para más información, podemos acceder al sitio oficial: http://pear.php.net/package/Image_Graph.

Figura 61. En el sitio oficial de *Image_Graph*, se incluyen ejemplos de uso listos para implementar en nuestros desarrollos.

IMAGICK

La extensión **imagick** (*Imagick Image Library*) permite crear y modificar archivos de imagen utilizando el API de **ImageMagick** (www.imagemagick.org), un software para edición de imágenes. Se trata de una librería orientada a objetos y es de carácter experimental. Esto significa que las clases, métodos y propiedades listadas pueden no funcionar en el futuro y ser removidas sin previo aviso. Podemos obtener más datos acerca de la extensión en el sitio oficial de PHP, en la dirección www.php.net/imagick y en <http://pecl.php.net/package/imagick>.



APERTURA

Para los desarrolladores de sistemas web, resulta interesante saber que GD no es una extensión ligada a un formato específico o a alguna versión del lenguaje de programación en particular. Por el contrario, trata de incluir más y más opciones en cada nueva versión disponible para los usuarios, algo que los desarrolladores reconocen y agradecen.

```
extension=php_imagick.dll
```

The screenshot shows a section of the PHP documentation titled "Image Processing (ImageMagick)". The left sidebar contains a navigation tree with categories like "PHP Manual", "Function Reference", and "Image Processing and Generation". Under "Image Processing and Generation", "ImageMagick" is listed. The main content area displays a hierarchical list of functions under "ImageMagick", including "adaptiveBlurImage", "adaptiveResizeImage", "adaptiveSharpenImage", "adaptiveThresholdImage", "addImage", "addNoiseImage", "affineTransformImage", "animateImages", "annotateImage", and "appendImage".

Figura 62. La extensión *imagick* está en plena etapa de desarrollo y se perfila como una opción muy valiosa en un futuro cercano.

Su uso no varía demasiado en relación con la forma tradicional de trabajo en PHP. En el siguiente ejemplo, generamos una miniatura de una imagen y la enviamos al navegador como salida:

```
<?php

$imagen = new Imagick('imagen.jpg');

$imagen->thumbnailImage(50, 0);

$salida = $imagen->getImageBlob();

$tipo = $imagen->getFormat();

header("Content-type: $tipo");
```

```
echo $salida;  
?>
```

El constructor **Imagick** recibe como argumento la ruta a una imagen. El método **thumbnailImage** recibe dos argumentos (ancho y alto) y, si alguno es igual a 0, se mantiene el aspecto original de la imagen (**aspect radio**). El método **getImageblob** recupera el contenido de la imagen, mientras que **getFormat** permite obtener el tipo de archivo (las últimas dos líneas del ejemplo utilizan estas informaciones para mostrar la imagen a través del navegador del usuario).

En el manual de PHP, encontraremos el listado completo de las opciones disponibles, aunque cabe destacar que se encuentran en revisión y es posible que no funcionen correctamente. La extensión **imagick** no utiliza la librería GD. ImageMagick brinda interfaces no sólo para PHP, sino también para un variado conjunto de lenguajes:

- **G2F** (Ada)
- **MagickCore** (C)
- **MagickWand** (C)
- **ChMagick** (Ch)
- **ImageMagickObject** (COM+)
- **Magick++** (C++)
- **JMagick** (Java)
- **L-Magick** (Lisp)
- **NMagick** (Neko/haXe)
- **MagickNet** (.NET)
- **PascalMagick** (Pascal)
- **PerlMagick** (Perl)
- **MagickWand** (PHP)
- **PythonMagick** (Python)
- **RMagick** (Ruby)
- **TclMagick** (Tcl/Tk)



VERSIONES DISPONIBLES

Debemos saber que según la versión de GD que hayamos seleccionado e instalado, las funcionalidades ofrecidas pueden variar y de esta forma afectar a nuestras aplicaciones. En la mayoría de los casos, esto no sucede, puesto que los cambios entre las versiones disponibles para los usuarios, por lo menos hasta ahora, no son muy radicales.

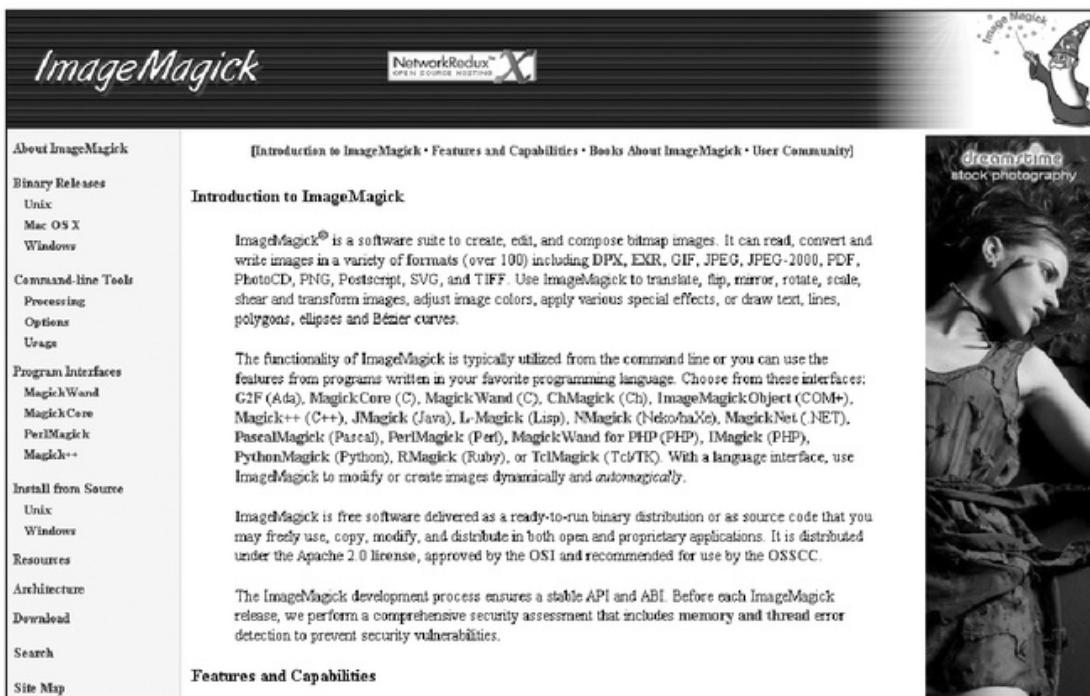


Figura 63. La aplicación **ImageMagick** admite funcionalidades avanzadas en lo relativo al tratamiento de archivos de imagen.

Es de uso libre (tiene una licencia compatible con **GPL**) y está disponible para la mayoría de los sistemas operativos.

... RESUMEN

En este capítulo observamos algunas de las aplicaciones que hacen uso de la librería GD para generar y manipular imágenes desde PHP: PHP Thumbnailer, phpThumb, JpGraph e Image_Graph, la opción de PEAR. Además, introdujimos la extensión imagick, una alternativa cada vez más utilizada en los desarrollos.



ACTIVIDADES

PREGUNTAS TEÓRICAS

- 1** ¿Por qué cree que GD es tan popular?

- 2** ¿Utiliza directamente la extensión GD en sus desarrollos? ¿Por qué?

- 3** ¿Por qué algunas versiones de GD no soportan el formato GIF?

- 4** ¿En qué casos utilizaría la caché para almacenar imágenes?

- 5** ¿Cuál es el estado actual de la extensión imagick?

EJERCICIOS PRÁCTICOS

- 1** Genere un álbum fotográfico utilizando PHP Thumbnailer.

- 2** Desarrolle un buscador de imágenes con phpThumb.

- 3** Implemente una encuesta online y grafique los resultados parciales y finales con Jp-Graph.

- 4** Modifique la aplicación anterior utilizando Image_Graph en lugar de JpGraph.

Servicios al lector

En este último apartado encontraremos un índice temático que nos ayudará a encontrar las palabras más significativas de esta obra y así acceder a los aspectos más importantes.

ÍNDICE TEMÁTICO

A

ABMC	140
ActiveRecord	138
addBlock	53
AD0db	135, 274, 280
Ajax	96, 256, 257
Apache	28, 113, 135
Asincronía	96
ASP	97, 106
Autocarga de clases	210
autoload_filters	70



B

blockExists	53
Bloques	42, 45
BSD	106

C

cache_handler_func	70
cache_lifetime	70
cache_modified_check	70
caching	70, 82, 270
CakePHP	135, 136, 140, 144, 150
Capas	116, 169, 170
Captcha	234, 235, 310, 366
Checkbox	74
Ciclos	72
Clases	19, 199, 203, 207, 210, 213
Cold Fusion	106

Comentarios	66
compile_check	70
compile_id	70
config_booleanize	70
config_overwrite	70
config_read_hidden	70
Constantes	19, 26, 203, 277, 355
Consultas preparadas	275
Controlador	134
CORBA	172
createTable	290
CRUDs	140
crypt	246
CSS	94, 95
Ctype	237
CURL	255

D

DB	296
DCOM	172
debugging	70, 140
default_template_handler_func	70
DHTML	94, 96
DocBlock	17
DocumentRoot	252



E

Encapsulación	203
Encriptación	226

error_reporting	26, 30
Excepciones	27, 34, 36



F	
Fines	346
Flexy	47, 311
G	
Generadores	147, 149
getBlockList	53
getCode	36

H	
Hash	241, 242
Herencia múltiple	213
hideBlock	46
Hosting	22
HTML_QuickForm	301, 302, 303

I	
Imagick	386, 387, 388
Inclusiones	19
Io	106
IT	40

J	
JavaDoc	16
JavaScript	19, 94
JpGraph	348, 349, 350, 351

M	
math	82
mcrypte	247

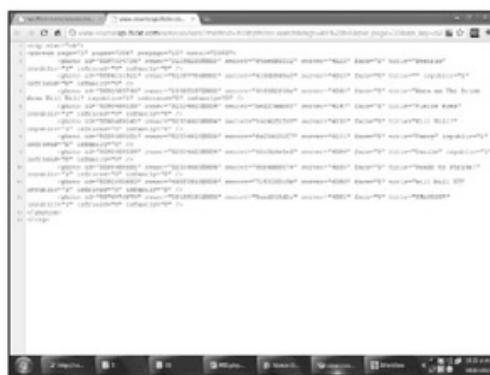
MD5	242, 250
MDB2	273, 274, 275
Metabase	274
Métodos abstractos	214
Métodos estáticos	215
Métodos	214
MIT	337, 338
Modelos	134, 142, 143

0

Options	24
Output	24

P

Pager	288, 333, 336
Parse	45
PATH	266
Pear	266, 270, 271



PEAR_Exception	27
PECL	268
Perl	16, 106, 300
Perror	31
PhD	23
PHP Documentor	15
PHP Thumbnailer	319, 320
PHP_CodeSniffer	24
PHPLib	274
placeholderExists	53
plugins_dir	70
P00	199
Postfiltros	81
Prefiltros	81

Protocolos	171, 188	setTemplate	42
R			
RAD	57	SHA-1	242
Reflexión	211, 328	T	
Register Globals	258	Templates	47
		Templateeyes	60, 91, 165
replaceBlock	43, 53	textformat	69
request_use_auto_globals	70	Thumbnails	326
S			
Safe mode	260	touchBlock	46, 48
Sajax	106, 107, 109	U	
secure_dir	70	UDDI	176, 177
security	70	Urlencode	231
security_settings	70	W	
setCallbackFunction	52	W3C	98, 99, 176
setCurrentBlock	49	Web 2.0	94
setRoot	42	WSDL	174



CLAVES PARA COMPRAR UN LIBRO DE COMPUTACIÓN

1 SOBRE EL AUTOR Y LA EDITORIAL

Revise que haya un cuadro "sobre el autor", en el que se informe sobre su experiencia en el tema. En cuanto a la editorial, es conveniente que sea especializada en computación.

2 PRESTE ATENCIÓN AL DISEÑO

Compruebe que el libro tenga guías visuales, explicaciones paso a paso, recuadros con información adicional y gran cantidad de pantallas. Su lectura será más ágil y atractiva que la de un libro de puro texto.

3 COMPARE PRECIOS

Suele haber grandes diferencias de precio entre libros del mismo tema; si no tiene el valor en tapa, pregunte y compare.

4 ¿TIENE VALORES AGREGADOS?

Desde un sitio exclusivo en la Red hasta un CD-ROM, desde un Servicio de Atención al Lector hasta la posibilidad de leer el sumario en la Web para evaluar con tranquilidad la compra, o la presencia de adecuados índices temáticos, todo suma al valor de un buen libro.

5 VERIFIQUE EL IDIOMA

No sólo el del texto; también revise que las pantallas incluidas en el libro estén en el mismo idioma del programa que usted utiliza.

6 REVISE LA FECHA DE PUBLICACIÓN

Está en letra pequeña en las primeras páginas; si es un libro traducido, la que vale es la fecha de la edición original.

 **usershop.redusers.com**
VISITE NUESTRO SITIO WEB

- » Vea información más detallada sobre cada libro de este catálogo.
- » Obtenga un capítulo gratuito para evaluar la posible compra de un ejemplar.
- » Conozca qué opinaron otros lectores.
- » Compre los libros sin moverse de su casa y con importantes descuentos.
- » Publique su comentario sobre el libro que leyó.
- » Manténgase informado acerca de las últimas novedades y los próximos lanzamientos.

TAMBIÉN PUEDE CONSEGUIR NUESTROS LIBROS EN KIOSCOS O PUESTOS DE PERIÓDICOS, LIBRERÍAS, CADENAS COMERCIALES, SUPERMERCADOS Y CASAS DE COMPUTACIÓN.



LLEGAMOS A TODO EL MUNDO VÍA »oCA * Y  **

* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

● usershop.redusers.com // ✉ usershop@redusers.com



Premiere + After Effects

Esta obra nos presenta un recorrido detallado por las aplicaciones audiovisuales de Adobe: Premiere Pro, After Effects y Soundbooth. Todas las técnicas de los profesionales, desde la captura de video hasta la creación de efectos, explicadas de forma teórica y práctica.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-9-3



Office 2010

En este libro aprenderemos a utilizar todas las aplicaciones de la suite, en su versión 2010. Además, su autora nos mostrará las novedades más importantes, desde los minigráficos de Excel hasta Office Web Apps, todo presentado en un libro único.

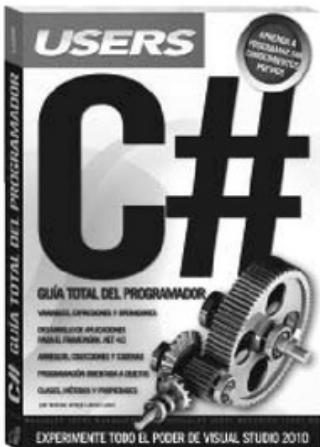
→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-26013-6-2



Excel Paso a Paso

En esta obra encontraremos una increíble selección de proyectos pensada para aprender, mediante la práctica, la forma de agilizar todas las tareas diarias. Todas las actividades son desarrolladas en procedimientos paso a paso de una manera didáctica y fácil de comprender.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-4-8



C#

Este libro es un completo curso de programación con C# actualizado a la versión 4.0. Ideal tanto para quienes desean migrar a este potente lenguaje, como para quienes quieran aprender a programar desde cero en Visual Studio 2010.

→ COLECCIÓN: MANUALES USERS
→ 400 páginas / ISBN 978-987-26013-5-5



200 Respuestas Seguridad

Esta obra es una guía básica que responde, en forma visual y práctica, a todas las preguntas que necesitamos contestar para conseguir un equipo seguro. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-26013-1-7



Funciones en Excel

Este libro es una guía práctica de uso y aplicación de todas las funciones de la planilla de cálculo de Microsoft. Desde las funciones de siempre hasta las más complejas, todas presentadas a través de ejemplos prácticos y reales.

→ COLECCIÓN: MANUALES USERS
→ 368 páginas / ISBN 978-987-26013-0-0

¡Léalo antes Gratis!

En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



Proyectos con Windows 7

En esta obra aprenderemos cómo aprovechar al máximo todas las ventajas que ofrece la PC. Desde cómo participar en las redes sociales hasta las formas de montar una oficina virtual, todo presentado en 120 proyectos únicos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-036-8



PHP 6

Este libro es un completo curso de programación de PHP en su versión 6.0. Un lenguaje que se destaca tanto por su versatilidad como por el respaldo de una amplia comunidad de desarrolladores, que lo convierten en un punto de partida ideal para quienes comienzan a programar.

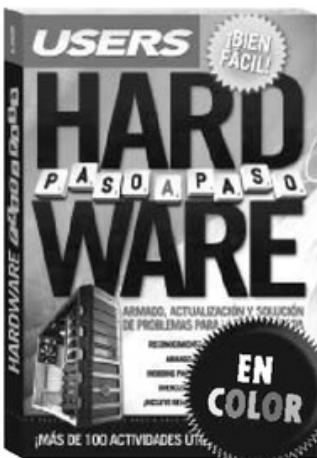
→ COLECCIÓN: MANUALES USERS
→ 368 páginas / ISBN 978-987-663-039-9



200 Respuestas: Blogs

Esta obra es una completa guía que responde a las preguntas más frecuentes de la gente sobre la forma de publicación más poderosa de la Web 2.0. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

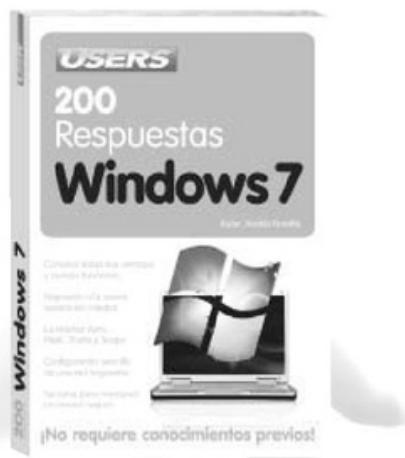
→ COLECCIÓN: 200 RESPUESTAS
→ 320 páginas / ISBN 978-987-663-037-5



Hardware paso a paso

En este libro encontraremos una increíble selección de actividades que abarcan todos los aspectos del hardware. Desde la actualización de la PC hasta el overclocking de sus componentes, todo en una presentación nunca antes vista, realizada íntegramente con procedimientos paso a paso.

→ COLECCIÓN: PASO A PASO
→ 320 páginas / ISBN 978-987-663-034-4



200 Respuestas: Windows 7

Esta obra es una guía básica que responde, en forma visual y práctica, a todas las preguntas que necesitamos conocer para dominar la última versión del sistema operativo de Microsoft. Definiciones, consejos, claves y secretos, explicados de manera clara, sencilla y didáctica.

→ COLECCIÓN: 200 RESPUESTAS
→ 320 páginas / ISBN 978-987-663-035-1



Office paso a paso

Este libro presenta una increíble colección de proyectos basados en la suite de oficina más usada en el mundo. Todas las actividades son desarrolladas con procedimientos paso a paso de una manera didáctica y fácil de comprender.

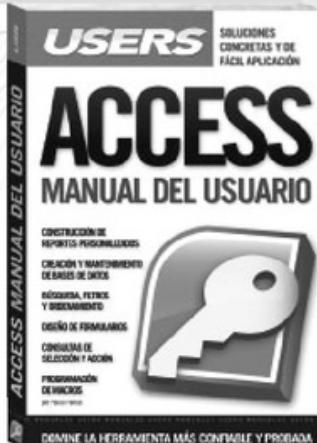
→ COLECCIÓN: PASO A PASO
→ 320 páginas / ISBN 978-987-663-030-6



101 Secretos de Hardware

Esta obra es la mejor guía visual y práctica sobre hardware del momento. En su interior encontraremos los consejos de los expertos sobre las nuevas tecnologías, las soluciones a los problemas más frecuentes, cómo hacer overclocking, modding, y muchos más trucos y secretos.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-029-0



Access

Este manual nos introduce de lleno en el mundo de Access para aprender a crear y administrar bases de datos de forma profesional. Todos los secretos de una de las principales aplicaciones de Office, explicados de forma didáctica y sencilla.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-025-2



Redes Cisco

Este libro permitirá al lector adquirir todos los conocimientos necesarios para planificar, instalar y administrar redes de computadoras. Todas las tecnologías y servicios Cisco, desarrollados de manera visual y práctica en una obra única.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-024-5



Proyectos con Office

Esta obra nos enseña a usar las principales herramientas de Office a través de proyectos didácticos y útiles. En cada capítulo encontraremos la mejor manera de llevar adelante todas las actividades del hogar, la escuela y el trabajo.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-023-8



Dreamweaver y Fireworks

Esta obra nos presenta las dos herramientas más poderosas para la creación de sitios web profesionales de la actualidad. A través de procedimientos paso a paso, nos muestra cómo armar un sitio real con Dreamweaver y Fireworks sin necesidad de conocimientos previos.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-022-1



Excel revelado

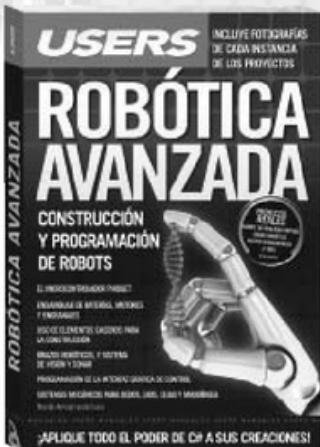
Este manual contiene una selección de más de 150 consultas de usuarios de Excel y todas las respuestas de Claudio Sánchez, un reconocido experto en la famosa planilla de cálculo. Todos los problemas encuentran su solución en esta obra imperdible.

→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-021-4



¡Léalo antes Gratis!

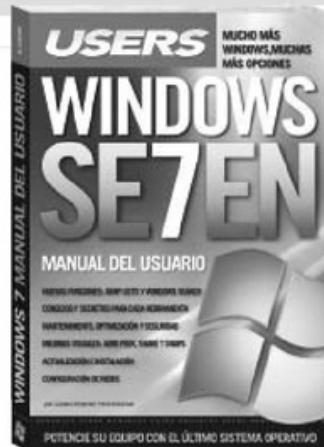
En nuestro sitio, obtenga GRATIS un capítulo del libro de su elección antes de comprarlo.



Robótica avanzada

Esta obra nos permitirá ingresar al fascinante mundo de la robótica. Desde el ensamblaje de las partes hasta su puesta en marcha, todo el proceso está expuesto de forma didáctica y sencilla para así crear nuestros propios robots avanzados.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-020-7



Windows 7

En este libro, encontraremos las claves y los secretos destinados a optimizar el uso de nuestra PC tanto en el trabajo como en el hogar. Aprenderemos a llevar adelante una instalación exitosa y a utilizar todas las nuevas herramientas que incluye esta versión.

→ COLECCIÓN: MANUALES USERS
→ 320 páginas / ISBN 978-987-663-015-3



De Windows a Linux

Esta obra nos introduce en el apasionante mundo del software libre a través de una completa guía de migración, que parte desde el sistema operativo más conocido: Windows. Aprenderemos cómo realizar gratuitamente aquellas tareas que antes hacíamos con software pago.

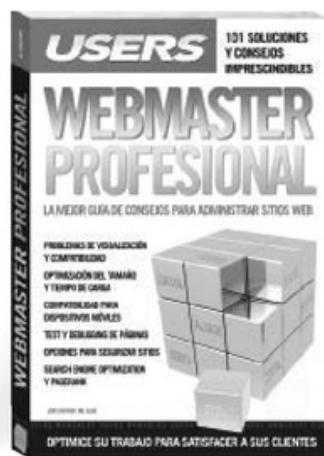
→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-013-9



Producción y edición de video

Un libro ideal para quienes deseen realizar producciones audiovisuales con bajo presupuesto. Tanto estudiantes como profesionales encontrarán cómo adquirir las habilidades necesarias para obtener una salida laboral con una creciente demanda en el mercado.

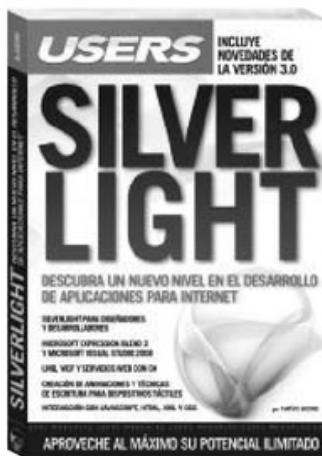
→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-012-2



Webmaster profesional

Esta obra explica cómo superar los problemas más frecuentes y complejos que enfrenta todo administrador de sitios web. Ideal para quienes necesiten conocer las tendencias actuales y las tecnologías en desarrollo que son materia obligada para dominar la Web 2.0.

→ COLECCIÓN: MANUALES USERS
→ 336 páginas / ISBN 978-987-663-011-5



Silverlight

Este manual nos introduce en un nuevo nivel en el desarrollo de aplicaciones interactivas a través de Silverlight, la opción multiplataforma de Microsoft. Quien consiga dominarlo creará aplicaciones visualmente impresionantes, acordes a los tiempos de la incipiente Web 3.0.

→ COLECCIÓN: MANUALES USERS
→ 352 páginas / ISBN 978-987-663-010-8

USERS

PRESENTA...

¡EL PRIMER EBOOK USERS!

Sí, ya podés leer Hackers al descubierto en tu PC, notebook, Amazon Kindle, iPad, en el celular...

**CONSEGUILO
DESDE CUALQUIER
PARTE DEL MUNDO**

**A UN PRECIO
INCREÍBLE**

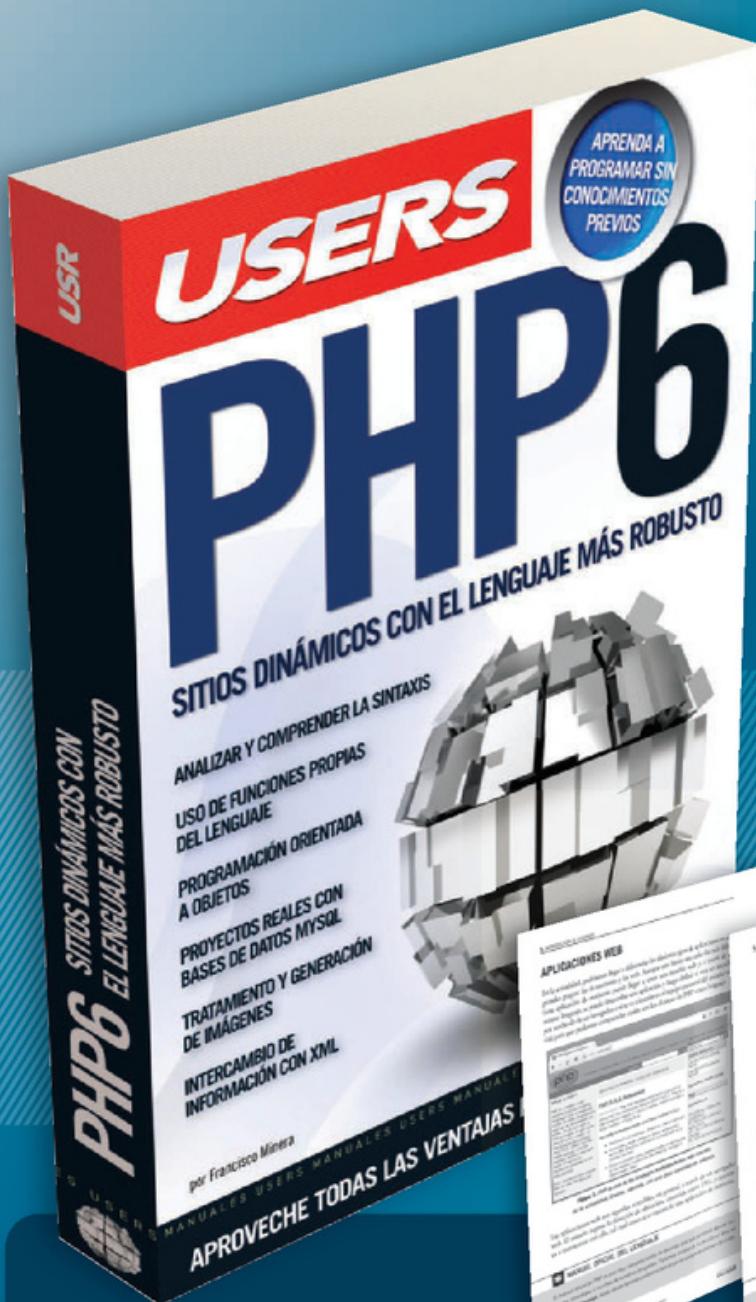
**¿QUÉ ESTÁS
ESPERANDO?**



**¡LEELO
DONDE
QUIERAS!**

INGRESA YA A USERSSHOP.REDUSERS.COM Y ENTERATE MÁS

**APRENDA A CREAR
SITIOS DINÁMICOS
CON EL LENGUAJE
MÁS ROBUSTO**



Este libro es un completo curso de programación de PHP en su versión 6.0. Un lenguaje que se destaca tanto por su versatilidad como por el respaldo de una amplia comunidad de desarrolladores, que lo convierten en un punto de partida ideal para quienes comienzan a programar.

- » MANUALES USERS
- » 368 PÁGINAS
- » ISBN 978-987-663-039-9



LLEGAMOS A TODO EL MUNDO VÍA  * Y  **

* SÓLO VÁLIDO EN LA REPÚBLICA ARGENTINA // ** VÁLIDO EN TODO EL MUNDO EXCEPTO ARGENTINA

 usershop.redusers.com //  usershop@redusers.com

CONTENIDO

1 | PLANEAMIENTO Y CONTROL DE PROYECTOS

Ciclo de vida de una aplicación - Documentación a través de PHPDocumentor - Etiquetas - Forma de implementación - Manejo y control de errores - Directivas - Configuración desde el archivo PHP.INI y desde SCRIPTS PHP - Excepciones - Captura y tratamiento

2 | TEMPLATES

Cuando utilizar templates - Sigma - La opción de PEAR - Bloques - Archivos externos - Smarty - Operadores - Clases - Archivos de configuración - Ciclos - Caché - Opciones para programadores y diseñadores

3 | WEB 2.0 / AJAX

Arquitectura cliente servidor - Interfaces AJAX - Usabilidad - Claves del modelo - El objeto XMLHttpRequest - Limitaciones - Alternativas - Sajax - Xajax - jPOP - Intercambio de datos con Json - XML

4 | RAD EN PHP

Desarrollo rápido de aplicaciones - Modelos, controladores y vistas - CakePHP - Configuración y modo de uso - Vinculación con la base de datos - Relaciones entre modelos - Templates

5 | SERVICIOS WEB

Capas, componentes y protocolos asociados - XML - RPC - SOAP - WSDL - UDDI - Implementaciones en PHP - Extensiones disponibles - Acceso a través de PEAR - REST

6 | ORIENTACIÓN A OBJETOS

POO en PHP - Clases - Tipos de acceso - Constantes - Captura de errores - Herencia - Clases abstractas - Propiedades y métodos estáticos

7 | SEGURIDAD EN APLICACIONES WEB

Filtrado de datos - Formularios - CAPTCHA - Bases de datos - Algoritmos de encriptación - Inclusión de archivos - Seguridad en aplicaciones AJAX - Funciones de línea de comandos - Variables globales y modo seguro

8 | PEAR

Comandos disponibles - Forma de utilización - Manejo y control de errores - Acceso a datos con MDB2 - Hojas de cálculo con Spreadsheet_Excel_Writer - Generación y manejo de formularios con HTML_QuickForm

9 | GENERACIÓN Y TRATAMIENTO DE GRÁFICOS

Librería GD - Manejo de imágenes con PHP Thumbnailer y phpThumb - Gráficos estadísticos con JpGraph - Tortas, Anillos, Barras, Leds, CAPTCHA, mapas HTML - Image_Graph La extensión Imagick

NIVEL DE USUARIO

PRINCIPIANTE

INTERMEDIO

AVANZADO

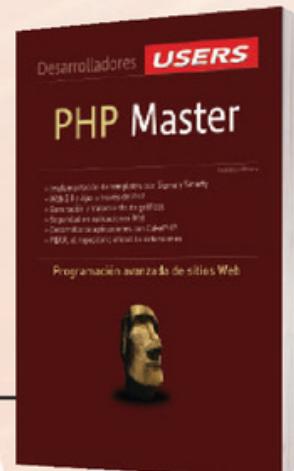
EXPERTO

PHP AVANZADO

DOMINE EL LENGUAJE MÁS CONFIABLE Y ESTABLE

Este libro brinda las herramientas necesarias para acercar al trabajo diario del desarrollador los avances más importantes incorporados en PHP 6. Destinado a programadores que tengan un conocimiento medio del lenguaje, en sus páginas se exponen las técnicas actuales que permiten potenciar el desarrollo de sitios web. Cada tecnología y herramienta es tratada en forma didáctica y acompañada de ejemplos prácticos concretos que ayudan a la comprensión.

El recorrido se realiza de la mano de Francisco Minera, un reconocido desarrollador de sistemas y plataformas para pequeñas y medianas empresas, y autor de varios libros sobre PHP, MySQL, XML y AJAX.



Versión completamente mejorada,
actualizada y ampliada de la obra
"PHP Master"

RedUSERS.com

En este sitio encontrará una gran variedad de recursos y software relacionado, que le servirán como complemento al contenido del libro. Además, tendrá la posibilidad de estar en contacto con los editores, y de participar del foro de lectores, en donde podrá intercambiar opiniones y experiencias.

Si desea más información sobre el libro puede comunicarse con nuestro Servicio de Atención al Lector: usershop@redusers.com

ADVANCED PHP

This book offers the necessary tools to be on top of this technology, and helps the developer to be up to date with the latest advances included in the most recent version of PHP.



MANUALES USERS MANUALES USERS MANUAL

PROGRAMACIÓN DE SITIOS WEB PROFESIONALES