CUANTICA

ASOCIACIÓN COOP. DE SERV. Y TECNOLOGÍA

Gerencia de Formación y Capacitación

INTRODUCCIÓN A LA PROGRAMACIÓN CON PHP

DAVID SUÁREZ PRESUTTI

GUILEVALDO PÉREZ HÓBRITZ

Prefacio

A mis estimados lectores, el desarrollo de aplicaciones web es, si no una tarea indispensable, altamente demandada en una economía globalizada que cada día depende más de los procesos eficientes de comunicación como factor catalizador del desarrollo sostenido de los entornos.

El objetivo del presente trabajo es introducir al programador de conocimiento intermedio al campo del desarrollo orientado a objetos con lenguajes web de alto desempeño y contribuir con un proceso de formación diseñado para expandir las perspectivas y capacidades técnicas de un colectivo de web másters que día a día adquieren consciencia del profundo cambio filosófico que está sufriendo el desarrollo web con la llegada de las páginas dinámicas, las aplicaciones del comercio y el ciclo de negocios completamente integrado a la internet, en todos y cada unos de sus procesos y formar en dicha masa un conjunto crítico de productores de oportunas soluciones a las desafiantes demandas de las operaciones económicas en línea.

Los conceptos y casos propuestos en el presente manual no obedecen a complejos recursos extraídos de bibliografías avanzadas, pero están constituidos por las experiencias de los autores a lo largo de su contacto con el campo del desarrollo de aplicaciones web en sectores como la industria automotriz , la banca y la construcción, entre otros y que de manera entusiasta hoy desean compartir y ofrecer a quienes se apunten en el presente programa de capacitación, acerca de las reseñas curriculares de los mismos, se puede citar su ficha en la comunidad de http://www.neurona.com

Nuestro objetivo en CUANTICA no es iniciar al aficionado al campo del diseño y la programación web, pero sí, poner a disposición del entusiasta un conjunto de herramientas didácticas lo suficientemente prácticas y entendibles para el participante, pero a la vez muy técnicas y aplicativas para los requerimientos del mercado actual.

Nuestra organización despierta día a día con un profundo deseo de servir, de ayudar, de impulsar y acercar al presente el desarrollo tecnológico de nuestras comunidades, nuestros pueblos, nuestras naciones. Queda de mi parte declararme a la plena, total e incondicional disposición de quienes hoy se inician en este apasionante campo y decir que todo en cuanto a mi alcance esté en aras de contribuir con el proceso de formación y mejoramiento técnico de los interesados haré.

David Suárez Presutti

1. Introducción a la programación en PHP

PHP es uno de los lenguajes de lado servidor más extendidos en la web. Nacido en 1994, se trata de un lenguaje de creación relativamente creciente que ha tenido una gran aceptación en la comunidad de desarrolladores web debido sobre todo a la potencia y simplicidad que lo caracterizan.

PHP permite acoplar sus pequeños fragmentos de código dentro de la página HTML y realizar determinadas acciones de una forma fácil y eficaz sin tener que generar programas desarrollados íntegramente en un lenguaje distinto al HTML. Por otra parte, y es aquí donde reside su característica más interesante con respecto a los lenguajes pensados para los CGI, y es que PHP ofrece un sinfín de funciones para la explotación de bases de datos de una manera llana, sin complicaciones.

Podría efectuarse la quizás radical comparación de decir que PHP y ASP son lenguajes parecidos en cuanto a potencia y dificultad si bien su sintaxis puede diferir sensiblemente. Algunas diferencias principales pueden, no obstante, mencionarse:

-PHP, aunque multiplataforma, ha sido concebido inicialmente para entornos UNIX y es en este sistema operativo donde se pueden aprovechar mejor sus prestaciones. ASP, siendo una tecnología Microsoft, esta orientado hacia sistemas Windows, especialmente NT.

-Las tareas fundamentales que puede realizar directamente el lenguaje son definidas en PHP como funciones mientras que ASP invoca más frecuentemente los objetos. Por supuesto, esto no es más que una simple cuestión de forma ya que ambos lenguajes soportan igualmente ambos procedimientos.

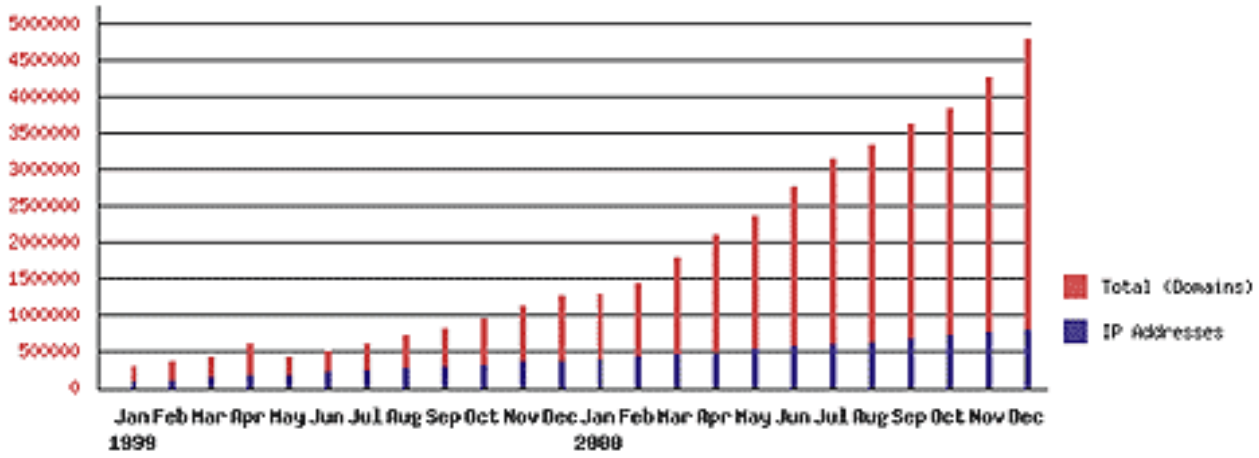
-ASP realiza numerosas tareas sirviéndose de componentes (objetos) que deben ser comprados (o programados) por el servidor a determinadas empresas especializadas. PHP presenta una filosofía totalmente diferente y, con una esencia más generosa, es progresivamente construido por colaboradores desinteresados que implementan nuevas funciones en nuevas versiones del lenguaje.

2. Breve historia de PHP

PHP es un lenguaje creado por una gran comunidad de personas. El sistema fue desarrollado originalmente en el año 1994 por Rasmus Lerdorf como un CGI escrito en lenguaje C que permitía la interpretación de un número limitado de comandos. El sistema fue denominado Personal Home Page Tools y adquirió relativo éxito gracias a que otras personas pidieron a Rasmus que les permitiese utilizar sus programas en sus propias páginas. Dada la aceptación del primer PHP y de manera adicional, su creador diseñó un sistema para procesar formularios al que le atribuyó el nombre de FI (Form Interpreter) y el conjunto de estas dos herramientas, sería la primera versión compacta del lenguaje: PHP/FI.

La siguiente gran contribución al lenguaje se realizó a mediados del año 1997 cuando se volvió a programar el analizador sintáctico, se incluyeron nuevas funcionalidades como el soporte a nuevos protocolos de Internet y el soporte a la gran mayoría de las bases de datos comerciales. Todas estas mejoras sentaron las bases de PHP versión 3. Actualmente PHP se encuentra en su versión 5, que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de las anteriores versiones. Algunas mejoras de esta nueva versión son su rapidez, gracias a que primero se compila y luego se ejecuta el código, mientras que antes se ejecutaba mientras se interpretaba el mismo, su mayor independencia del servidor web creando versiones de PHP nativas para más plataformas y un API más elaborado y con más funciones.

Fig. 1.- Gráfica del número de dominios y direcciones IP que utilizan PHP. Estadística de Netcraft.



En el último año, el número de servidores que utilizan PHP se ha incrementado a una tasa exponencial, logrando situarse cerca de los 5 millones de sitios y 800.000 direcciones IP, lo que ha convertido a PHP en una tecnología popular. Esto es debido, entre otras razones, a que PHP es el complemento ideal para que el tándem Linux-Apache sea compatible con la programación del lado del servidor de sitios web. Gracias a la aceptación que ha logrado, y los grandes esfuerzos realizados por una creciente comunidad de colaboradores para implementarlo de la manera más óptima, podemos asegurar que el lenguaje se convertirá en un estándar que compartirá los éxitos augurados al conjunto de sistemas desarrollados en código abierto.

3. Tareas principales del PHP

Poco a poco el PHP se va convirtiendo en un lenguaje que facilita la ejecución de una infinita gama de tareas. En un principio diseñado para realizar poco más que un contador y un libro de visitas, PHP ha experimentado en poco tiempo una verdadera y acelerada evolución y, a partir de sus funciones, en estos momentos se pueden realizar una multitud de actividades útiles para el desarrollo del web:

3.1 Funciones de correo electrónico

Puede con una facilidad asombrosa enviarse un e-mail a una persona o lista parametrizando toda una serie de aspectos tales como el e-mail de procedencia, asunto, persona a responder, etc.

Otras funciones menos frecuentes pero de indudable utilidad para gestionar correos electrónicos son incluidas en su librería.

3.2 Gestión de bases de datos

Resulta difícil concebir un sitio web actual, dinámico, eficaz, interactivo, potente y rico en contenido que no es gestionado por una base de datos. El lenguaje PHP ofrece interfaces para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft, a partir de las cuales se podrá editar el contenido de un sitio web con absoluta sencillez.

3.3. Gestión de archivos

Crear, borrar, mover, modificar...cualquier tipo de operación razonable puede ser realizada a partir de una amplia librería de funciones para la gestión de archivos por PHP. También se facilita la transferencia de archivos a través de FTP a partir de sentencias en cualquier código, protocolo para el cual PHP ha previsto también gran cantidad de funciones.

3.4 Tratamiento de imágenes

Evidentemente resulta mucho más sencillo utilizar Photoshop para el tratamiento de imágenes pero...¿Y si se hace presente la necesidad de tratar miles de imágenes enviadas por los visitantes de un sitio web?

La verdad es que puede resultar muy complejo y extenso uniformar en tamaño y formato miles de imágenes recibidas día tras día. Todo esto puede ser también automatizado eficazmente mediante PHP.

También puede parecer útil el crear botones dinámicos, es decir, botones en los que se aplique el mismo diseño y solo cambien el texto. Puede, por ejemplo crearse un botón haciendo una única llamada a una función en la que se introduce el estilo del botón y el texto a introducir obteniendo automáticamente el botón deseado.

A partir de la librería de funciones graficas puede hacerse esto y mucho más.

Muchas otras funciones pensadas para Internet (tratamiento de cookies, accesos restringidos, comercio electrónico, banking...) o para propósito general (funciones matemáticas, explotación de cadenas, de fechas, corrección ortográfica, compresión de archivos...) son realizadas por este lenguaje. A esta inmensa librería cabe ahora añadir todas las funciones personales que un usuario puede crear por necesidades propias y que luego son reutilizadas en otros sitios y todas aquellas intercambiadas u obtenidas en foros o sitios especializados.

Como puede verse, las posibilidades que se presentan son sorprendentemente innumerables.

4. Instalación de PHP en un servidor propio

Como todo lenguaje de lado servidor, PHP, requiere de la instalación de un servidor en el PC del programador para poder trabajar en un entorno local. Este modo de trabajo resulta a todas luces más práctico que fijar los archivos vía FTP en el servidor y ejecutarlos desde Internet.

Así pues, antes comenzar a crear programas propios en PHP, es necesario:

- Convertir el ordenador en un servidor. Esto se hace instalando uno de los varios servidores disponibles para el sistema operativo del PC que se vaya a utilizar.

- Introducir en el equipo servidor los archivos que le permitirán la comprensión del PHP.

Estos archivos pueden ser descargados, en su versión más actual, de la página oficial de PHP; http://www.php.net/downloads.php

Para conocer la forma de instalar PHP sobre cada servidor de cada sistema operativo puede remitirse al apartado de documentación al cual puede accesarse a través de la página oficial de PHP; http://www.php.net/docs.php, donde se dispone de un manual en HTML de rápida consulta y un enorme manual en PDF de casi 1000 páginas traducido al castellano donde explican minuciosamente y entre otras cosas, los pasos a seguir para cada caso particular. No obstante, en el presente manual se ofrecen diversas ayudas para configurar PHP en las plataformas más habituales.

La elección del programa servidor tendrá mucho que ver con el sistema operativo del cual se disponga en el PC. Las siguientes serían algunas posibilidades de sistemas operativos y soluciones que funcionan bien.

4.1. Sistemas Operativos

4.1.1 Windows 95/98

Si se está trabajando en Windows 95 o Windows 98 y para desarrolladores principiantes, podría ser recomendable utilizar el servidor Personal Web Server. En este caso se requerirá:

- Personal Web Server de Microsoft como servidor el cual os sirve además para el aprendizaje en ASP. Se adjunta al presente manual el complemento guías de instalación y configuración, ver Capítulo I.

Es necesario señalar que, para el caso de PHP en PWS (Personal Web Server), además de todo lo dicho en capítulo de instalación, es importante al crear el directorio virtual permitir la ejecución de scripts validando la caja correspondiente.

En Windows 95/98 también puede utilizarse el servidor Apache y puede que sea una opción todavía más completa que la de utilizar PWS. A continuación se amplía más sobre ello.

4.1.2 Windows ME y XP Home edition

No se han efectuado pruebas de funcionalidad PHP en estas plataformas, pero en principio no tienen compatibilidad con Personal Web Server, por lo que se debe optar por otro servidor.

Otra posibilidad para los usuarios de Windows en general es instalar Apache como servidor web lo cual puede resultar ventajoso con respecto al uso del PWS ya que PHP está principalmente diseñado para correr en este servidor. Esto quiere decir que, aunque en principio todo debería funcionar correctamente sobre ambos servidores, es posible que algún bug no corregido haga fallar uno de nuestros scripts si trabajamos para con un servidor cuyas actualizaciones son menos frecuentes y detalladas.

Apache ha sido especialmente pensado para plataformas Unix-Linux, aunque recientemente, con la Apache 2.0, han desarrollado una versión específica para Windows.

En el complemento guías de Instalación y Configuración podrá disponerse de un artículo para aprender a configurar PHP sobre Apache en Windows, como CGI y también como módulo de Apache.

4.1.3 Windows NT, Windows 2000 y XP en sus versiones Profesional y Server

Para estos sistemas se tendrán dos (02) posibilidades muy interesantes, ya que se podrá instalar PHP sobre Internet Information Server o sobre Apache con todas las garantías. Si hubiese que recomendar una de las dos opciones, se sugeriría optar por Apache debido a que, como se mencionaba previamente, PHP está pensado para trabajar sobre dicho servidor web. Podría ser interesante IIS en el caso de que se desee ejecutar ASP y PHP sobre el mismo servidor, ya que, en principio, Apache no es compatible con ASP.

4.1.4 Unix - Linux

Hay que decir, no obstante, que las mejores prestaciones de este lenguaje son obtenidas trabajando en entorno Unix o Linux y con un servidor Apache, la combinación más corriente en la mayoría de los servidores de Internet que trabajan con PHP.

Referencia: En el complemento de Guía de Instalación y Configuración, a suministrarse en la etapa II del presente módulo, en los capítulos III, IV, V, VI, VII Y VIII se dispone de artículos que pueden ser una buena referencia para la instalación de PHP.

Configuración de PHP con Apache en Windows
Configuración de PHP como módulo de Apache, también en Windows
Instalación del Personal Web Server
Instalación de IIS en Windows XP profesional
Directorio de Apache

4.1.5 Deliberación

En cualquier caso, para fines de desarrollo en local, puede centrarse atención, en un principio, de trabajar con cualquier sistema operativo. Solamente en casos de programación realmente avanzada se podrán confrontar problemas relacionados con el sistema operativo utilizado o el servidor en el que se hagan correr las páginas. Hay que pensar también que, en casos puntuales para los que el PC pueda quedarse escaso de recursos, se pueden realizar las pruebas directamente en el servidor donde se aloja el sitio, el cual será muy probablemente, como ya se ha dicho dicho, un Unix o Linux funcionando con Apache.

4.2 Configuración de PHP con Apache en Windows, como CGI

El presente artículo trata de cómo **configurar PHP y Apache** para que trabajen conjuntamente en un sistema **Windows**. Además, este artículo asume que hay un servidor Apache configurado en el Windows, y que funciona correctamente.

Existen dos formas de configurar PHP para trabajar con Apache, instalar como un módulo o instalar como un CGI. En este artículo se apreciará cómo instalarlo como CGI, aunque se dispone de un espacio dedicado para instalar PHP como módulo en Apache en el Capítulo IV del Complemento Guía de Instalación y configuración.

4.2.1 Secuencia de pasos para instalar PHP como un CGI

En primer lugar, hay que descargar PHP desde la página de php.net. Existen dos versiones, una que tiene un instalador, y otra que es un fichero ZIP. Será necesario descargar esta última.

Una vez descargado, hay que descomprimirlo dentro de una carpeta, esta no tiene que estar bajo el árbol de directorios de Apache. El artículo asumirá que se descomprime dentro de la carpeta C:\PHP. Comprobar que los contenidos del archivo ZIP no quedan en un subdirectorio de la carpeta C:\PHP, sino directamente en dicha carpeta.

Dentro de la carpeta c:\PHP se encuentra un archivo llamado PHP4ts.dll, hay que mover dicho archivo dentro de la carpeta: c:\windows\system ó c:\winnt\system

A continuación, dentro de la carpeta c:\php se encuentra un archivo llamado php.ini-recomended. Hay que copiar este dentro de la carpeta c:\Windows, y renombrarlo a php.ini.

En este fichero se encuentra toda la configuración de PHP, y las modificaciones en la configuración de PHP (mostrar Errores, variables globales etc.), se encuentra dentro del mismo.

Es muy recomendable cambiar la directiva display_errors que por defecto esta en OFF, y ponerla en ON, para poder ver los errores que se producen en las páginas durante el desarrollo. Para un servidor en producción en conveniente dejarla en OFF.

Una vez se han hecho estos cambios, queda indicarle al Apache, donde se encuentra instalado el PHP, para ello hay que editar el fichero httpd.conf que se encuentra dentro de la carpeta conf, en la carpeta de instalación del apache (por defecto c:\archivos de programa\apache group\apache2\conf)

Abrir el fichero, y situarse al final del mismo, y escribir las siguientes líneas:

```
ScriptAlias /php/ "c:/php/"
AddType application/x-httpd-php .php
Action application/x-httpd-php "/php/php.exe"
```

En ellas se indica donde se encuentra el ejecutable de php, y lo asocia a los ficheros .php que

se encuentren dentro de apache.

A continuación reiniciar el servidor Apache, y de tal manera se concluiría el procedimiento.

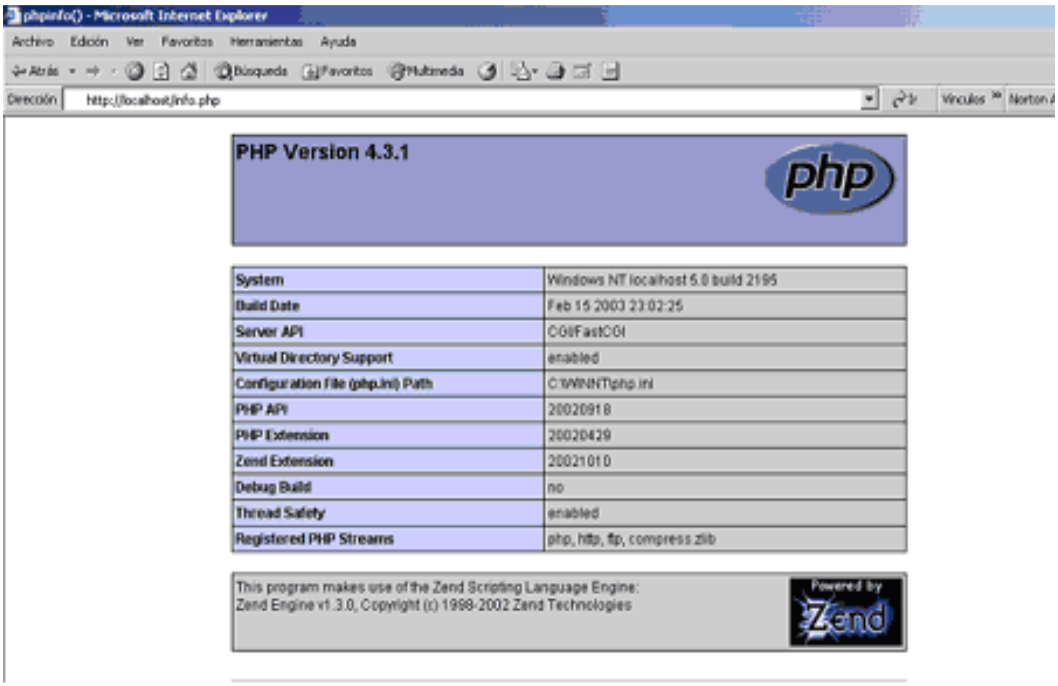
Posteriormente, se sugiere a efectos de probar la nueva instalación, crear un archivo php con el siguiente contenido:

<? phpinfo();?>

Luego lo almacenarlo dentro de la carpeta raíz de documentos del Apache (por defecto c:\archivos de programa\apache group\apache2\htdocs), con un nombre terminado en .php, por ejemplo info.php

Para ejecutarlo, a través de un navegador, escribir la dirección http://localhost/info.php. Debería aparecer una pantalla como la que se muestra a continuación.

Fig. 2.- Ventana de Explorador con Elementos y Datos de Configuración de Instalación de PHP como CGI.



Si se aprecia una ventana como la anterior, se ha ejecutado el proceso de manera correcta y se dispone de PHP perfectamente instalado en el servidor Apache.

4.3 Configuración de PHP como modulo de Apache en Windows

En este artículo se procede a explicar cómo instalar PHP como módulo de Apache 2.0 en un sistema Windows. Para las pruebas se ha utilizado Windows XP, pero seguro que con otros sistemas el proceso será muy parecido, aunque, en todo caso, se indicarán las diferencias documentadas en el sitio de PHP.

Anteriormente se había explicado la instalación de PHP como un CGI, aunque en la página de PHP desaconsejan esta opción, puesto que genera de grabes problemas de seguridad. Además, PHP instalado como módulo de Apache resulta mucho más rápido que como CGI.

Referencia: Supóngase que el servidor de páginas web Apache 2.0 está instalado en el sistema operativo. No obstante, para los que no lo tengan, les referimos al adjunto Complemento Guía de instalación y configuración de, Capítulo IX dedicado a la configuración de Apache.

4.3.1 Descargar y descomprimir PHP

El primer paso consiste en descargar la última versión de PHP. Puede hacerse desde la página oficial de PHP, en la sección de descargas. http://www.php.net/downloads.php Debe elegirse la versión "zip package" que contiene todas las funcionalidades de PHP y el módulo necesario para instalarlo en Apache.

Una vez descargado el paquete comprimido en .zip de PHP es necesario descomprimirlo en disco duro. Puede utilizarse el directorio raíz del disco duro para descomprimir los archivos. En ese caso, se creará un directorio llamado "php-4.3.1-Win32" que dependerá del directorio raíz.

Se recomienda cambiar el nombre del directorio creado a "c:\php". En todo caso, se advierte en el sitio oficial de PHP sobre no colocar ningún nombre de directorio que contenga espacios, pues algún servidor web puede dar problemas. Por ejemplo, cuidado con instalar PHP en un directorio como este "c:\archivos de programa\php", pues en la ruta existirán directorios con espacios.

4.3.2 Copia de las DLL

A continuación se debe adoptar conciencia sobre la necesidad de copiar en el directorio de sistema una serie de librerías (.dll), que se hallarán en el directorio sapi de la instalación de PHP, la ruta "c:\php\sapi"

El mencionado directorio de sistema puede variar de unas versiones a otras de Windows. En Windows XP, el directorio de sistema donde deben copiarse las dll, es:

"C:\WINDOWS\system32". En Windows 9x/ME, el directorio sería "C:\Windows\System" y en Windows NT/2000 sería el directorio "C:\WINNT\System32" o bien, "C:\WINNT40\System32".

Nota: no se deben mezclar las DLL de diversas versiones de PHP, porque de lo contrario, podría causar problemas.

4.3.3 Definir un archivo php.ini

Otro archivo que debe copiarse, esta vez en el directorio Windows, es el php.ini, que guarda las opciones de configuración definidas para PHP. En la distribución de PHP se incluyen dos archivos php.ini que pueden ser utilizados directamente en el sistema. Estos dos archivos se llaman "php.ini-dist" y "php.ini-recommended" y contienen unas opciones típicas de configuración de PHP. Se recomienda utilizar "php.ini-recommended", porque viene optimizado para obtener los mejores niveles de seguridad. En cualquier caso, puede editarse en cualquier momento el contenido del archivo para modificar la configuración de PHP según propio criterio, gusto o necesidades.

Para definir el php.ini debe hacerse una copia del archivo de configuración escogido ("php.ini-dist" o "php.ini-recommended") y renombrarlo como el "php.ini". Posteriormente debe copiarse en la carpeta Windows, que en sistemas 9x/ME/XP es "c:\windows" y en sistemas NT/2000 suele ser "c:\WINNT", o bien "c:\WINNT40".

4.3.4 Editar httpd.conf

Posteriormente deberá editarse el archivo de configuración de Apache, llamado "httpd.conf" que está en el directorio "conf" de la instalación de Apache. También puede encontrarse un acceso directo para editar este archivo accediendo a Inicio - Programas - Apache HTTP Server - Configure Apache HTTP Server - Edit httpd.conf configuration file.

Deben añadirse dos (02) líneas de configuración del módulo de Apache.

```
LoadModule php4_module C:\php\sapi\php4apache2.dll
AddType application/x-httpd-php .php
```

El lugar adecuado para añadir esas líneas es en el bloque de carga de módulos, que podemos encontrar si buscamos por el texto "LoadModule". Podemos añadir las líneas de carga del módulo PHP después de la carga de los otros módulos que vienen ya configurados en archivo httpd.conf de Apache.

Si no se ha instalado PHP en el directorio c:\php, deben editarse las líneas a colocar en el httpd.conf para colocar la ruta correcta al directorio donde está la librería php4apache2.dll.

4.3.5 Culminación de Procedimiento

Antes de concluir y probar si PHP se ha instalado correctamente, es necesario copiar un dll en el directorio sapi. Concretamente, el dll "php4ts.dll", que puede encontrarse en el directorio de instalación de PHP, es la que debe copiarse al directorio sapi, "c:\php\sapi".

> **Nota:** Esta acción no viene documentada en el manual de PHP, aunque sí no es llevada a cabo no funcionará.
>
> El error que se obtiene al tratar de arrancar el Apache es:
> Syntax error on line 173 of C:/Archivos de programa/Apache Group/Apache2/conf/httpd.conf: Cannot load C:/php/sapi/php4apache2.dll into server: No se puede encontrar el módulo especificado.

Otra configuración que puede aplicarse al archivo httpd.conf es definir también como documento por defecto el archivo index.php en el servidor Apache. El documento por defecto es generalmente index.html, pero lo habitual si se va a programar con PHP es que también se necesite definir index.php como documento a mostrar si no se indica otro documento del directorio al que se está accediendo.

El documento por defecto se define con la variable DirectoryIndex. Quedará una definición como esta: `DirectoryIndex index.html index.html.var index.php`

4.3.6 Prueba de Operatividad de PHP Post – Instalación

Para concluir, puede crearse una página de prueba de PHP, que colocaremos en nuestro directorio de publicación de Apache, generalmente llamado htdocs, que se aloja dentro del directorio donde se ha instalado Apache, algo como "C:\Archivos de programa\Apache Group\Apache2\htdocs"

Podemos crear un archivo llamado, por ejemplo, "prueba.php", en el que colocaremos dentro el siguiente código:

```
<?
phpinfo()
?>
```

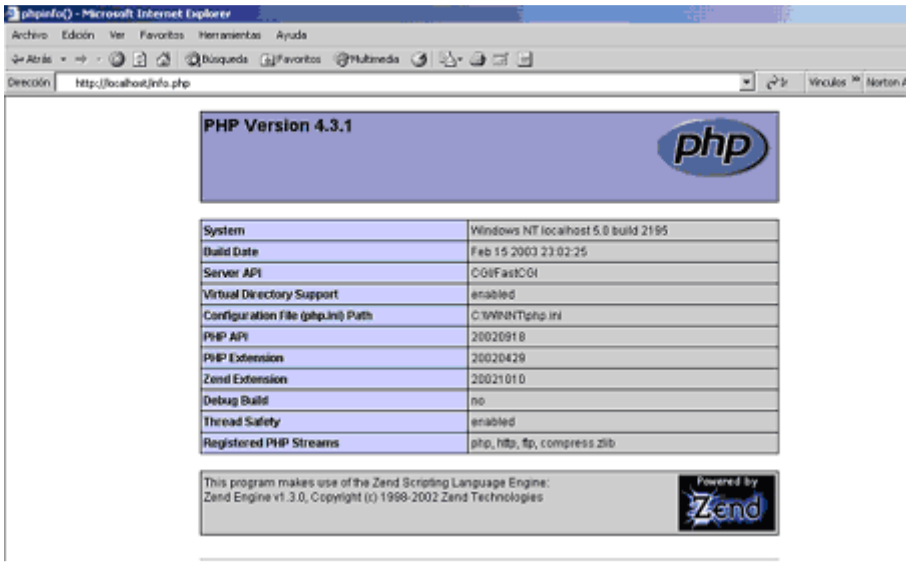
Esta función simplemente creará una página de muestra de las configuraciones definidas para PHP en ese servidor.

Para acceder al archivo creado desde nuestro explorador, escribiremos en la barra de direcciones esta URL:

http://localhost/prueba.php

Debería aparecer un resultado como el de la siguiente imagen.

Fig. 3.- Ventana de Explorador con Elementos y Datos de Configuración de Instalación de PHP como Módulo de Apache.



4.4 Instalación de MySQL en Windows

Uno de los puntos fuertes de las páginas en PHP es la posibilidad de explotar bases de datos mediante funciones de una simplicidad y potencia muy notables. Estas bases de datos pueden servir a un sitio web para almacenar contenidos de una forma sistemática, lo cual permita clasificarlos, buscarlos y editarlos rápida y fácilmente.

Una base de datos es sencillamente un conjunto de tablas en las que se almacenan distintos registros (titulares de cuentas bancarias y saldos disponibles, registro histórico de transacciones, artículos de una tienda virtual, proveedores o clientes de una empresa, películas en cartelera en el cine...). Estos registros son catalogados en función de distintos parámetros que los caracterizan y que presentan una utilidad a la hora de clasificarlos. Así, por ejemplo, los artículos de una tienda virtual podrían catalogarse a partir de distintos campos como puede ser un código de barras, número de referencia, nombre del artículo, descripción, precio, proveedor, etc.

La aplicación de bases de datos más difundida con el tandem UNIX-Apache es sin duda MySQL. Como para el caso de Apache, una versión para Windows está disponible y puede ser descargada gratis desde http://www.mysql.com.

Su puesta a punto no entraña mucha dificultad. Una vez instalado el programa podrán ejecutarse órdenes o comandos en modo MS-DOS. Para ello debe abrirse una ventana MS-DOS y situarse en el directorio *bin* de *mysql*. En este directorio se encuentran los archivos ejecutables. Aquí habrá que encontrar un archivo llamado *mysqld*. En el caso de la versión más actual durante la redacción de este artículo este archivo es llamado mysqld-shareware. Una vez ejecutado este archivo se podrá ejecutar el siguiente: *mysql*.

Ahora en este punto se apreciará cómo un mensaje de bienvenida aparece en la pantalla pantalla. En estos momentos se está dentro de la base de datos. A partir de ahí se podrán realizar todo tipo de operaciones por sentencias SQL.

No se ahondará en una explicación pormenorizada del funcionamiento de esta base de datos. Se sugiere como referencia el manual de SQL a partir del cual se puede tener una idea muy práctica de las sentencias necesarias para la creación y edición de las tablas. También existe una documentación extensa en inglés en el directorio *Docs* de MySQL. A modo de resumen, aquí se proponen además las operaciones más básicas que, combinadas al manual de SQL pueden dar solución a gran parte de los casos que se presenten:

Instrucción	Descripción
Show databases;	Muestra el conjunto de bases de datos presentes en el servidor
Use nombre_de_la_base	Determina la base de datos sobre la que vamos a trabajar
Create Database nombre_de_la_base;	Crea una nueva bd con el nombre especificado
Drop Database nombre_de_la_base;	Elimina la base de datos del nombre especificado
Show tables;	Muestra las tablas presentes en la base de datos actual
Describe nombre_de_la_tabla;	Describe los campos que componen la tabla
Drop Table nombre_de_la_tabla;	Borra la tabla de la base de datos
Load Data Local Infile "archivo.txt" Into Table nombre_de_la_tabla;	Crea los registros de la tabla a partir de un fichero de texto en el que separamos por tabulaciones todos los campos de un mismo registro.
Quit	Salir de MySQL

Para evitar el tener que editar tablas directamente sobre archivos de texto, puede resultar muy práctico usar cualquier otra base de datos con un editor y exportar a continuación la tabla en un archivo de texto configurado para dejar tabulaciones entre cada campo. Esto es posible en Access por ejemplo aplicando el botón derecho sobre la tabla que se desea convertir y eligiendo la opción exportar. Una ventana de dialogo aparecerá en la que se ordenará guardar el archivo en tipo texto. El paso siguiente será elegir un formato delimitado por tabulaciones sin cualificador de texto.

Otra posibilidad que puede resultar muy práctica y que evita trabajar continuamente componiendo órdenes es servirse de programas en PHP o Perl ya existentes y descargables en la red. El más popular sin duda es phpMyAdmin. Este tipo de scripts son ejecutados desde un navegador y pueden ser por tanto albergados en el servidor o empleados en local para, a partir de ellos, administrar MySQL de una forma menos compleja.

Asimismo, dentro del directorio bin de MySQL, puede encontrarse una pequeña aplicación llamada MySqlManager. Se trata de una interface windows, más agradable a la vista y al uso que la obtenida ejecutando el archivo *mysql*. En este caso, las sentencias SQL deben realizarse sin el punto y coma final.

5. Introducción a la sintaxis PHP

PHP se escribe dentro de la propia página web, junto con el código HTML y, como para cualquier otro tipo de lenguaje incluido en un código HTML, en PHP es necesario especificar cuáles son las partes constitutivas del código escritas en este lenguaje. Esto se hace, como en otros casos, delimitando el código por etiquetas. Pueden utilizarse distintos modelos de etiquetas en función de preferencias y costumbres. Hay que tener sin embargo en cuenta que no necesariamente todas están configuradas inicialmente y que otras, sólo están disponibles a partir de una determinada versión (3.0.4.).

Estos modos de abrir y cerrar las etiquetas son:

```
<?       y     ?>
<%       y     %>
<?php              y     ?>
<script languaje="php">
```

Este ultimo modo está principalmente aconsejado a aquellos que acostumbren a trabajar con editores web de bajo performance y nivel, como lo es el caso de Microsoft Front Page, ya que usando cualquier otro tipo de etiqueta, se corre el riesgo de que la aplicación la borre de manera arbitraria, esto debido a que se trata de un código incomprensible para ella.

El modo de funcionamiento de una página PHP, a grandes rasgos, no difiere del clásico para una página dinámica de lado servidor: El servidor va a reconocer la extensión correspondiente a la página PHP (phtml, php, php4,...) y antes de enviarla al navegador va a encargarse de interpretar y ejecutar todo aquello que se encuentre entre las etiquetas correspondientes al lenguaje PHP. El resto, lo enviara sin más ya que, asumirá que se trata de código HTML absolutamente comprensible por el navegador.

Otra característica general de los scripts en PHP es la forma de separar las distintas instrucciones. Para hacerlo, hay que cerrar o concluir cada instrucción con un punto y coma ";". Para la última expresión, la que va antes del cierre de etiqueta, este formalismo no es necesario.

Se incluye también en este capitulo la sintaxis de comentarios. Un comentario, para aquellos que no lo sepan, es una frase o palabra que se incluye en el código para comprenderlo más fácilmente al volverlo a leer un tiempo después y que, por supuesto, el ordenador tiene que ignorar ya que no va dirigido a él sino al desarrollador. Los comentarios tienen una gran utilidad ya que es muy fácil olvidarse del funcionamiento de un script programado un tiempo atrás y resulta muy útil si se desea hacer rápidamente comprensible un determinado código a otra persona.

Pues bien, la forma de incluir estos comentarios es variable dependiendo si se desea escribir una línea o más. Obsérvese esto con un primer ejemplo de script:

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

5.1 Scripts de Comentarios

5.1.1 Ejecutar script de Comentario

Si se usa doble barra (//) o el símbolo # pueden introducirse comentarios de una línea. Mediante /* y */ creamos comentarios multilínea. Por supuesto, nada nos impide de usar estos últimos en una sola línea.

Es prudente en esta oportunidad adelantar que las variables en PHP se definen anteponiendo un símbolo de dólar ($) y que la instrucción *echo* sirve para sacar en pantalla lo que hay escrito a continuación.

Debe recordarse que todo el texto insertado en forma de comentario es completamente ignorado por el servidor. Resulta importante acostumbrarse a dejar comentarios, es algo que se agradece con el tiempo.

5.2 Manejo de Variables en PHP

En el manual de páginas dinámicas se introdujo el concepto de variable. En el capítulo anterior se comentó que para PHP las variables son definidas anteponiendo el símbolo dólar ($) al

nombre de la variable que se está definiendo. Dependiendo de la información que contenga, una variable puede ser considerada de uno u otro tipo:

Variables numéricas Almacenan cifras		
Enteros	$entero=2002;	Números sin decimales
Real	$real=3.14159;	Números con o sin decimal

Variables alfanuméricas Almacenan textos compuestos de números y/o cifras		
Cadenas	Almacenan variables alfanuméricas	$cadena="Hola amigo";

Tablas Almacenan series de informaciones numéricas y/o alfanuméricas		
Arrays	Son las variables que guardan las tablas	$sentido[1]="ver"; $sentido[2]="tocar"; $sentido[3]="oir"; $sentido[4]="gusto"; $sentido[5]="oler";

Objetos
Se trata de conjuntos de variables y funciones asociadas. Presentan una complejidad mayor que las variables vistas hasta ahora pero su utilidad es más que interesante.

A diferencia de otros lenguajes, PHP posee una gran flexibilidad a la hora de operar con variables. En efecto, cuando se define una variable asignándole un valor, el ordenador le atribuye un tipo. Si por ejemplo se define una variable entre comillas, la variable será considerada de tipo cadena:

```
$variable="5"; //esto es una cadena
```

Sin embargo si se pide en el script realizar una operación matemática con esta variable, no se obtendrá un mensaje de error sino que la variable cadena será asimilada a numérica:

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

Este script dará como resultado "8". La variable cadena ha sido asimilada en entero (aunque su tipo sigue siendo cadena) para poder realizar la operación matemática. Del mismo modo, se puede operar entre variables tipo entero y real. No deben surgir preocupaciones, PHP se

encarga durante la ejecución de interpretar el tipo de variable necesario para el buen funcionamiento del programa.

Sin embargo, en contraste, hay que tener cuidado en no cambiar mayúsculas por minúsculas ya que, en este sentido, PHP es sensible. Conviene por lo tanto trabajar ya sea siempre en mayúsculas o siempre en minúsculas para evitar este tipo de malentendidos a veces muy difíciles de localizar.

5.2.1 Variables asignadas por referencia

En PHP también pueden asignarse variables por referencia. En ese caso no se les asigna un valor, sino otra variable, de tal modo que las dos variables comparten espacio en memoria para el mismo dato.

La notación para asignar por referencia es colocar un "&" antes del nombre de la variable.

```
<?php
$foo = 'Bob'; // Asigna el valor 'Bob' a $foo
$bar = &$foo; // Referencia $foo vía $bar.
$bar = "Mi nombre es $bar"; // Modifica $bar...
echo $foo; // $foo también se modifica.
echo $bar;
?>
```

Esto dará como resultado la visualización dos veces del string "Mi nombre es Bob". Algo como:

Mi nombre es BobMi nombre es Bob

5.3. Cambio del tipo de las variables en PHP

PHP no requiere que se indique el tipo que va a contener una variable, sino que lo deduce del valor que se le asigne a la variable. Asimismo, se encarga de actualizar automáticamente el tipo de la variable cada vez que se le asigne un nuevo valor.

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

Nota: Se excluyen en este caso el cambio de variables a tipo Array porque la sintaxis puede resultar ambigua al expresar ese código, es decir, puede darse el caso de que una línea de código pueda significar dos cosas.

```
$a = "1";
//$a es una cadena
$a[0] = "f";
//¿Estamos editando el índice de la cadena o forzando a array?
```

5.3.1 Forzado

En cualquier caso, puede forzarse una variable para que cambie de tipo con la función setType().

```
setType($variable,"nuevo_tipo");
```

la función setType() actualiza el tipo de $variable a "nuevo_tipo" y devuelve un booleano indicando si hubo éxito o no en la conversión.

Entre "nuevo_tipo" se tienen:

· "integer"
· "double"
· "string"
· "array"
· "object"

También puede hacerse que una variable se comporte como un tipo determinado forzándola, de la misma manera a como se hace en el lenguaje C.

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

Los forzados permitidos son:

· (int), (integer) - fuerza a entero (integer)
· (real), (double), (float) - fuerza a doble (double)
· (string) - fuerza a cadena (string)
· (array) - fuerza a array (array)
(object) - fuerza a objeto (object)

5.4 Variables de sistema en PHP

Dada su naturaleza de lenguaje de lado servidor, PHP es capaz de brindar acceso a toda una serie de variables que informan sobre el servidor y sobre el cliente. La información de estas variables es atribuida por el servidor y en ningún caso es posible modificar sus valores directamente mediante el script. Para hacerlo es necesario influir directamente sobre la propiedad que definen. Existen multitud de variables de este tipo, algunas sin utilidad aparente y otras realmente interesantes y con una aplicación directa para un sitio web comercial. Aquí se enumeran algunas de estas variables y la información que aportan:

Variable	Descripción
$HTTP_USER_AGENT	Informa principalmente sobre el sistema operativo y tipo y versión de navegador utilizado por el internauta. Su principal utilidad radica en que, a partir de esta información, podemos re direccionar nuestros usuarios hacia páginas optimizadas para su navegador o realizar cualquier otro tipo de acción en el contexto de un navegador determinado.
$HTTP_ACCEPT_LANGUAGE	Devuelve la o las abreviaciones de la lengua considerada como principal por el navegador. Esta lengua o lenguas principales pueden ser elegidas en el menú de opciones del navegador. Esta variable resulta también extremadamente útil para enviar al internauta a las páginas escritas en su lengua, si es que existen.
$HTTP_REFERER	Indica la URL desde la cual el internauta ha tenido acceso a la página. Muy interesante para generar botones de "Atrás" dinámicos o para crear nuestros propios sistemas de estadísticas de visitas.
$PHP_SELF	devuelve una cadena con la URL del script que está siendo ejecutado. Muy interesante para crear botones para recargar la página.
$HTTP_GET_VARS	Se trata de un array que almacena los nombres y contenidos de las variables enviadas al script por URL o por formularios GET
$HTTP_POST_VARS	Se trata de un array que almacena los nombres y contenidos de

	las variables enviadas al script por medio de un formulario POST
$HTTP_COOKIES_VARS	Se trata de un array que almacena los nombres y contenidos de las cookies. Veremos qué son más adelante.
$PHP_AUTH_USER	Almacena la variable usuario cuando se efectúa la entrada a páginas de acceso restringido. Combinado con $PHP_AUTH_PW resulta ideal para controlar el acceso a las páginas internas del sitio.
$PHP_AUTH_PW	Almacena la variable password cuando se efectúa la entrada a páginas de acceso restringido. Combinado con $PHP_AUTH_USER resulta ideal para controlar el acceso a las páginas internas del sitio.
$REMOTE_ADDR	Muestra la dirección IP del visitante.
$DOCUMENT_ROOT	Nos devuelve el path físico en el que se encuentra alojada la página en el servidor.
$PHPSESSID	Guarda el identificador de sesión del usuario. Veremos más adelante en qué consisten las sesiones.

No todas estas variables están disponibles en la totalidad de servidores o en determinadas versiones de un mismo servidor. Además, algunas de ellas han de ser previamente activadas o definidas por medio de algún acontecimiento. Así, por ejemplo, la variable $HTTP_REFERER no estará definida a menos que el internauta acceda al script a partir de un enlace desde otra página.

5.5. Variables De Tipo Global

A partir de PHP 4.1.0, se dispone de un conjunto de variables de tipo array que mantienen información del sistema, llamadas súper-globales porque se definen automáticamente en un ámbito global.

Estas variables hacen referencia a las mismas que se accedían antes por medio de los arrays del tipo $HTTP_*_VARS. Éstas todavía existen, aunque a partir de PHP 5.0.0 se pueden desactivar con la directiva register_long_arrays.

La lista de estas variables, extraída directamente de la documentación de PHP es la siguiente:

5.5.1 $GLOBALS

Contiene una referencia a cada variable disponible en el espectro de las variables del script. Las llaves de esta matriz son los nombres de las variables globales. $GLOBALS existe dese PHP 3.

5.5.2 $_SERVER

Variables definidas por el servidor web ó directamente relacionadas con el entorno en donde el script se está ejecutando. Análoga a la antigua matriz $HTTP_SERVER_VARS (la cual está todavía disponible, aunque no se use).

5.5.3 $_GET

Variables proporcionadas al script por medio de HTTP GET. Análoga a la antigua matriz $HTTP_GET_VARS (la cual está todavía disponible, aunque no se use).

5.5.4 $_POST
Variables proporcionadas al script por medio de HTTP POST. Análoga a la antigua matriz $HTTP_POST_VARS (la cual está todavía disponible, aunque no se use).

5.5.5 $_COOKIE
Variables proporcionadas al script por medio de HTTP cookies. Análoga a la antigua matriz $HTTP_COOKIE_VARS (la cual está todavía disponible, aunque no se use).

5.5.6 $_FILES
Variables proporcionadas al script por medio de la subida de ficheros vía HTTP . Análoga a la antigua matriz $HTTP_POST_FILES (la cual está todavía disponible, aunque no se use). Vea también Subiendo ficheros por método POST para más información.

5.5.7 $_ENV
Variables proporcionadas al script por medio del entorno. Análoga a la antigua matriz $HTTP_ENV_VARS (la cual está todavía disponible, aunque no se use).

5.5.8 $_REQUEST
Variables proporcionadas al script por medio de cualquier mecanismo de entrada del usuario y por lo tanto no se puede confiar en ellas. La presencia y el orden en que aparecen las variables en esta matriz son definidos por la directiva de configuración variables_order. Esta matriz no tiene un análogo en versiones anteriores a PHP 4.1.0. Vea también import_request_variables().

5.5.9 $_SESSION
Variables registradas en la sesión del script. Análoga a la antigua matriz $HTTP_SESSION_VARS (la cual está todavía disponible, aunque no se use). Vea también la sección Funciones para el manejo de sesiones para más información.

5.6. Tablas o Arrays en PHP

Un tipo de variable que ya se ha descrito pero puede ser relativamente complicado a asimilar con respecto a la mayoría son los arrays. Un array es una variable que está compuesta de varios elementos cada uno de ellos catalogado dentro de ella misma por medio de una clave.

En capítulos anteriores se ponía el ejemplo de un array llamado sentido que contenía los distintos sentidos del ser humano:

```
$sentido[1]="ver";
$sentido[2]="tocar";
$sentido[3]="oir";
$sentido[4]="gustar";
$sentido[5]="oler";
```

En este caso este array cataloga sus elementos, comúnmente llamados valores, por números. Los números del 1 al 5 son por lo tanto las claves y los sentidos son los valores asociados. Nada nos impide emplear nombres (cadenas) para clasificarlos. Lo único que deberá hacerse es encerrarlos entre comillas:

```
<?
$moneda["espana"]="Peseta";
$moneda["francia"]="Franco";
$moneda["usa"]="Dolar";
?>
```

Otra forma de definir idénticamente este mismo array y que puede ayudar para la creación de arrays más complejos es la siguiente sintaxis:

```
<?
$moneda=array("espana"=> "Peseta","francia" => "Franco","usa" => "Dolar");
?>
```

Una forma muy práctica de almacenar datos es mediante la creación de arrays multidimensionales (tablas). Pongamos el ejemplo siguiente: Se desea almacenar dentro de una misma tabla el nombre, moneda y lengua hablada en cada país. Para hacerlo puede emplearse un array llamado país que vendrá definido por estas tres características (claves). Para crearlo, debería escribirse una expresión del mismo tipo que la vista anteriormente en la que se mantiene un array dentro del otro. Este proceso de incluir una instrucción dentro de otra se llama anidar y es muy corriente en programación:

```
<?
$pais=array
(
"espana" =>array
  (
  "nombre"=>"España",
  "lengua"=>"Castellano",
  "moneda"=>"Peseta"
  ),
"francia" =>array
  (
  "nombre"=>"Francia",
  "lengua"=>"Francés",
  "moneda"=>"Franco"
  )
);
echo $pais["espana"]["moneda"] //Saca en pantalla: "Peseta"
?>
```

Antes de entrar en el detalle de este pequeño script, deben comentarse algunos puntos referentes a la sintaxis. Como puede verse, en esta secuencia de script, no se ha introducido punto y coma ";" al final de cada línea. Esto es simplemente debido a que lo que se ha escrito puede ser considerado como una sola instrucción. En realidad, es el programador quien decide cortarla en varias líneas para, así, facilitar su lectura. La verdadera instrucción acabaría una vez definido completamente el array y es precisamente ahí donde se ha colocado el único punto y coma. Por otra parte, puede observarse cómo se ha jugado con el tabulador para

separar unas líneas más que otras del principio. Esto también se hace por cuestiones de claridad, ya que facilita ver qué partes del código están incluidas dentro de otras. Es importante acostumbrarse a escribir de esta forma del mismo modo que a introducir los comentarios ya que la claridad de los scripts es fundamental a la hora de depurarlos. Un poco de esfuerzo a la hora de crearlos puede ahorrar muchas horas al momento de corregirlos o modificarlos en el futuro.

Pasando ya al comentario del programa, como puede apreciarse, éste permite almacenar tablas y, a partir de una simple petición, visualizarlas en pantalla.

Lo particularmente interesante es que la utilidad de los arrays no concluye en éste punto, sino que también pueden utilizarse toda una serie de funciones creadas para ordenarlos por criterio alfabético directo o inverso, por claves, contar el numero de elementos que componen el array además de la facilidad de moverse por dentro de él hacia delante o atrás.

Muchas son las funciones propuestas por PHP para el tratamiento de arrays, no se profundizará en este capítulo en una descripción de las mismas. Sólo se incluirá esta resumida tabla que puede ser complementada, si necesario, con *Manual Avanzado* suministrado en este curso en el módulo dedicado a desarrollo dinámico bajo php.

Función	Descripción
array_values (mi_array)	Lista los valores contenidos en mi_array
asort(mi_array) y arsort(mi_array)	Ordena por orden alfabético directo o inverso en función de los valores
count(mi_array)	Proporciona el numero de elementos de un array
ksort(mi_array) y krsort(mi_array)	Ordena por orden alfabético directo o inverso en función de las claves
list ($variable1, $variable2...)=mi_array	Asigna cada una variable a cada uno de los valores del array
next(mi_array), prev(mi_array), reset(mi_array) y end(mi_array)	Nos permiten movernos por dentro del array con un puntero hacia delante, atras y al principio y al final.
each(mi_array)	Nos da el valor y la clave del elemento en el que nos encontramos y mueve al puntero al siguiente elemento.

5.6. Trabajo con tablas o Arrays en PHP

Se procederá en éste capítulo a analizar varios ejemplos de trabajo con arrays (arreglos, vectores, matrices o tablas en castellano) en PHP que ilustrará racionalmente el funcionamiento de algunas de las funciones de arrays más populares que trae consigo PHP.

5.6.1 Modificar el número de elementos de un array

Ahora se procede a observar varios ejemplos mediante los cuales los arrays pueden aumentar o reducir el número de casillas disponibles.

5.6.1.1 Reducir el tamaño de un array

5.6.1.1.1 array_slice()

Para disminuir el número de casillas de un arreglo tenemos varias funciones. Entre ellas, array_slice() se utiliza cuando se quiere recortar algunas casillas del arreglo, sabiendo los índices de las casillas que se desean conservar.

Recibe tres parámetros. El array, el índice del primer elemento y el número de elementos a tomar, siendo este último parámetro opcional.

En el ejemplo siguiente se tiene un array con cuatro nombres propios. En la primera ejecución de array_slice() se está indicando que se desean tomar todos los elementos desde el índice 0 (el principio) hasta un número total de 3 elementos.

El segundo array_slice() indica que se tomen todos los elementos a partir del índice 1 (segunda casilla).

```
<?
$entrada = array ("Miguel", "Pepe", "Juan", "Julio", "Pablo");

//modifico el tamaño
$salida = array_slice ($entrada, 0, 3);
//muestro el array
foreach ($salida as $actual)
   echo $actual . "<br>";

echo "<p>";

//modifico otra vez
$salida = array_slice ($salida, 1);
//muestro el array
foreach ($salida as $actual)
   echo $actual . "<br>";
?>
```

Tendrá como salida:

Miguel
Pepe
Juan

Pepe
Juan

5.6.1.1.2 array_shift()

Esta función extrae el primer elemento del array y lo devuelve. Además, acorta la longitud del array eliminando el elemento que estaba en la primera casilla. Siempre hace lo mismo, por tanto, no recibirá más que el array al que se desea eliminar la primera posición.

En el código siguiente se tiene el mismo vector con nombres propios y se ejecuta dos veces la función array_shift() eliminando un elemento en cada ocasión. Se imprimen los valores que devuelve la función y los elementos del array resultante de eliminar la primera casilla.

```
<?
$entrada = array ("Miguel", "Pepe", "Juan", "Julio", "Pablo");
```

```
//quito la primera casilla
$salida = array_shift ($entrada);
//muestro el array
echo "La función devuelve: " . $salida . "<br>";
foreach ($entrada as $actual)
   echo $actual . "<br>";

echo "<p>";

//quito la primera casilla, que ahora sería la segunda del array original
$salida = array_shift ($entrada);
echo "La función devuelve: " . $salida . "<br>";
//muestro el array
foreach ($entrada as $actual)
   echo $actual . "<br>";
?>
```

Da como resultado:

La función devuelve: Miguel
Pepe
Juan
Julio
Pablo

La función devuelve: Pepe
Juan
Julio
Pablo

5.6.1.1.3 unset()

Se utiliza para destruir una variable dada. En el caso de los arreglos, se puede utilizar para eliminar una casilla de un array asociativo (los que no tienen índices numéricos sino que su índice es una cadena de caracteres).

Obsérvese el siguiente código para conocer cómo definir un array asociativo y eliminar luego una de sus casillas.

```
<?
$estadios_futbol = array("Barcelona"=> "Nou Camp","Real Madrid" => "Santiago Bernabeu","Valencia" => "Mestalla","Real Sociedad" => "Anoeta");

//mostramos los estadios
foreach ($estadios_futbol as $indice=>$actual)
   echo $indice . " -- " . $actual . "<br>";

echo "<p>";

//eliminamos el estadio asociado al real madrid
unset ($estadios_futbol["Real Madrid"]);

//mostramos los estadios otra vez
foreach ($estadios_futbol as $indice=>$actual)
echo $indice . " -- " . $actual . "<br>";
?>
```

La salida será la siguiente:

Barcelona -- Nou Camp
Real Madrid -- Santiago Bernabeu
Valencia -- Mestalla
Real Sociedad -- Anoeta

Barcelona -- Nou Camp
Valencia -- Mestalla
Real Sociedad -- Anoeta

5.6.1.2 Aumentar el tamaño de un array

Se tienen también a disposición varias funciones que pueden ayudar a aumentar el número de casillas de un arreglo.

5.6.1.2.1 array_push()

Inserta al final del array una serie de casillas que se le indiquen por parámetro. Por tanto, el número de casillas del array aumentará en tantos elementos como se hayan indicado en el parámetro de la función. Devuelve el número de casillas del array resultante.

Veamos este código donde se crea un arreglo y se añaden luego tres nuevos valores.

```
<?
$tabla = array ("Lagartija", "Araña", "Perro", "Gato", "Ratón");

//aumentamos el tamaño del array
array_push($tabla, "Gorrión", "Paloma", "Oso");

foreach ($tabla as $actual)
   echo $actual . "<br>";
?>
```

Da como resultado esta salida:

Lagartija
Araña
Perro
Gato
Ratón
Gorrión
Paloma
Oso

5.6.1.2.2 array_merge()

Ahora se analizará cómo unir dos arrays utilizando la función array_merge(). A ésta se le pasan dos o más arrays por parámetro y devuelve un arreglo con todos los campos de los vectores pasados.

En este código de ejemplo se crearon tres (03) arrays y luego fueron unidos mediante la función array_merge()

```
<?
$tabla = array ("Lagartija", "Araña", "Perro", "Gato", "Ratón");
$tabla2 = array ("12","34","45","52","12");
$tabla3 = array ("Sauce","Pino","Naranjo","Caracas","Perro","34");
```

```
//aumentamos el tamaño del array
$resultado = array_merge($tabla, $tabla2, $tabla3);

foreach ($resultado as $actual)
   echo $actual . "<br>";
?>
```

Da como resultado:

Lagartija
Araña
Perro
Gato
Ratón
12
34
45
52
12
Sauce
Pino
Naranjo
Caracas
Perro
34

Una última cosa. También pueden introducirse nuevas casillas en un arreglo por los métodos habituales de asignar las nuevas posiciones en el array a las casillas que se necesiten.

En arrays normales se haría así:

```
$tabla = array ("Sauce","Pino","Naranjo");
$tabla[3]="Algarrobo";
```

En arrays asociativos:

```
$estadios_futbol = array("Valencia" => "Mestalla","Real Sociedad" => "Anoeta");
$estadios_futbol["Barcelona"]= "Nou Camp";
```

Se verán más adelante otras posibilidades del trabajo con arrays.

5.7. Cadenas

Una de las variables más corrientes a las que deberá hacerse frente en la mayoría de los scripts son las cadenas, que no son más que información de carácter no numérico (textos, por ejemplo).

Para asignar a una variable un contenido de este tipo, se escribirá entre comillas, dando lugar a declaraciones de este tipo:

```
$cadena="Esta es la información de la variable"
```

Si se quiere ver en pantalla el valor de una variable o bien un mensaje cualquiera usaremos la instrucción *echo* como ya lo hemos visto anteriormente:

```
echo $cadena //sacaría "Esta es la información de la variable"
echo "Esta es la información de la variable" //daría el mismo resultado
```

Podemos yuxtaponer o concatenar varias cadenas poniendo para ello un punto entre ellas:

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

También pueden introducirse variables dentro de una cadena, lo cual puede ayudar mucho en el desarrollo de un script. Lo que se verá no es el nombre, sino el valor de la variable:

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

La pregunta que puede plantearse ahora es ¿Qué hacer entonces para que en vez del valor "55" salga el texto "$a"? En otras palabras, cómo se hace para que el símbolo $ no defina una variable sino que sea tomado tal cual. Esta pregunta es tanto más interesante cuanto que en algunos scripts este símbolo debe ser utilizado por una simple razón comercial (pago en dólares por ejemplo) y si se escribe tal cual, el ordenador va a pensar que lo que viene detrás es una variable siendo que no lo es.

Pues bien, para introducir éste y otros caracteres utilizados por el lenguaje dentro de las cadenas y no confundirlos, lo que hay que hacer es escribir una contra-barra (slash) delante:

Carácter	Efecto en la cadena
\$	Escribe dólar en la cadena
\"	Escribe comillas en la cadena
\\	Escribe contra-barra en la cadena
\8/2	Escribe 8/2 y no 4 en la cadena

Además, existen otras utilidades de esta contra-barra que permiten introducir en un documento HTML determinados eventos:

Carácter	Efecto en la cadena
\t	Introduce una tabulación en nuestro texto
\n	Cambio de línea
\r	Retorno de carro

Estos cambios de línea y tabulaciones tienen únicamente efecto en el código y no en el texto ejecutado por el navegador. En otras palabras, si se quiere que el texto ejecutado cambie de línea ha de introducirse un *echo "
"* y no *echo "\n"* ya que este ultimo sólo cambia de línea en el archivo HTML creado y enviado al navegador cuando la página es ejecutada en el servidor. La diferencia entre estos dos elementos puede ser fácilmente comprendida mirando el código fuente producido al ejecutar este script:

```
<HTML>
<HEAD>
<TITLE>cambiolinea.php</TITLE>
</HEAD>
<BODY>
<?
echo "Hola, \n sigo en la misma línea ejecutada pero no en código fuente.<br>Ahora cambio
de línea ejecutada pero continuo en la misma en el código fuente."
?>
</BODY>
</HTML>
```

Al solicitar Código fuente del Navegador lo que se observaría seria lo siguiente:

```
<HTML>
<HEAD>
<TITLE>cambiolinea.php</TITLE>
</HEAD>
<BODY>
Hola,
 sigo en la misma línea ejecutada pero no en código fuente.<br>Ahora cambio de línea
ejecutada pero continuo en la misma en el código fuente.</BODY>
</HTML>
```

Las cadenas pueden asimismo ser tratadas por medio de funciones de todo tipo. Se verán estas funciones más adelante con más detalle. Tan sólo debe retenerse que existen muchas posibles acciones que pueden realizarse sobre ellas: dividirlas en palabras, eliminar espacios sobrantes, localizar secuencias, remplazar caracteres especiales por su correspondiente en HTML o incluso extraer las etiquetas META de una página web.

5.7. Funciones en PHP

En el previo módulo de páginas dinámicas del presente curso se introdujo y analizó el concepto de función. Se dijo que la función podría ser definida como un conjunto de instrucciones que explotan ciertas variables para realizar una tarea más o menos elemental.

PHP basa su eficacia principalmente en este tipo de elemento. Una gran librería que crece constantemente, a medida que nuevas versiones van surgiendo, es complementada con las funciones de propia cosecha, dando como resultado un sinfín de recursos que son aplicados por una simple llamada.

Las funciones integradas en PHP son muy fáciles de utilizar. Tan sólo ha de realizarse la llamada de la forma apropiada y especificar los parámetros y/o variables necesarios para que la función realice su tarea.

Lo que puede parecer ligeramente más complicado, pero que resulta sin lugar a dudas muy práctico, es crear funciones propias. De una forma general, el programador podría crear funciones propias para conectarlo a una base de datos o crear los encabezados o etiquetas meta de un documento HTML. Para una aplicación de comercio electrónico el programador podría crear por ejemplo funciones de cambio de una moneda a otra o de cálculo de los impuestos a añadir al precio de artículo. En definitiva, es interesante crear funciones para la mayoría de acciones más o menos sistemáticas que se realizan en programas.

Aquí se suministrará el ejemplo de creación de una función que, llamada al comienzo del script, crea el encabezado de un documento HTML y coloca el titulo que se desea a la página:

```
<?
function hacer_encabezado($titulo)
{
$encabezado="<html>\n<head>\n\t<title>$titulo</title>\n</head>\n";
echo $encabezado;
}
?>
```

Esta función podría ser llamada al principio de todas las páginas de la siguiente forma:

```
$titulo="Mi web";
hacer_encabezado($titulo);
```

De esta forma se automatiza el proceso de creación del documento. Se podría por ejemplo incluir en la función otras variables que ayudasen a construir la etiquetas meta y de esta forma, con un esfuerzo mínimo, se crearían los encabezados personalizados para cada una de las páginas desarrolladas por el programador. De este mismo modo se hace posible crear cierres de documento o formatos diversos para textos como si se tratase de hojas de estilo que tendrían la ventaja de ser reconocidas por todos los navegadores.

Por supuesto, la función ha de ser definida dentro del script ya que no se encuentra integrada en PHP sino que la ha creado el progamador. Esto en realidad no impone ninguna dificultad ya que puede ser incluida desde un archivo en el que se irán almacenando las definiciones de las

funciones que el programador vaya creando o recopilando.

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

```
require("libreria.php") o include("libreria.php")
```

En resumen, la línea quedaría así:

Se tendría un archivo libreria.php como sigue:

```
<?
//función de encabezado y colocación del titulo
Function hacer_encabezado($titulo)
{
$encabezado="<html>\n<head>\n\t<title>$titulo</title>\n</head>\n";
echo $encabezado;
}
?>
```

Por otra parte se tendríamos un script principal página.php (por ejemplo):

```
<?
include("libreria.php");
$titulo="Mi Web";
hacer_encabezado($titulo);
?>
<body>
El cuerpo de la página
</body>
</html>
```

Pueden introducirse todas las funciones que se van encontrando dentro de un mismo archivo pero resulta muchísimo más ventajoso ir clasificándolas en distintos archivos por temática: Funciones de conexión a bases de datos, funciones comerciales, funciones generales, etc. Esto ayudará al programador a poder localizarlas antes para corregirlas o modificarlas, permitirá también cargar únicamente el tipo de función que se necesite para el script sin recargar éste en exceso además de permitir utilizar un determinado tipo de librería para varios sitios webs distintos.

También puede resultar muy práctico el utilizar una nomenclatura sistemática a la hora de nombrarlas: Las funciones comerciales podrían ser llamadas com_loquesea, las de bases de datos bd_loquesea, las de tratamiento de archivos file_loquesea. Esto nos permitirá reconocerlas enseguida cuando leamos el script sin tener que recurrir a nuestra oxidada memoria para descubrir su utilidad.

Como puede verse, la tarea del programador puede en algunos casos parecerse a la de un coleccionista. Hay que ser paciente y metódico y al final, a base de trabajo propio, intercambio

y tiempo puedes llegar a poseerse una rica gama de funcionalidades.

5.7.1 Ejemplo de función

en esta oportunidad se procederá a ver un ejemplo de creación de funciones en PHP. Se trata de hacer una función que recibe un texto y lo escribe en la página con cada carácter separado por "-". Es decir, si recibe "hola" debe escribir "h-o-l-a" en la página web.

La manera de realizar esta función será recorrer el string, caracter a caracter, para imprimir cada uno de los caracteres, seguido de el signo "-". Se recorrerá el string con un bucle for, desde el carater 0 hasta el número de caracteres total de la cadena.

El número de caracteres de una cadena se obtiene con la función predefinida en PHP strlen(), que recibe el string entre paréntesis y devuelve el número de los caracteres que tenga.

```
<html>
<head>
   <title>funcion 1</title>
</head>

<body>

<?
function escribe_separa($cadena){
   for ($i=0;$i<strlen($cadena);$i++){
      echo $cadena[$i];
      if ($i<strlen($cadena)-1)
         echo "-";
   }
}

escribe_separa ("hola");
echo "<p>";
escribe_separa ("Texto más largo, a ver lo que hace");
?>
</body>
</html>
```

La función que se ha creado se llama escribe_separa y recibe como parámetro la cadena que hay que escribir con el separador "-". El bucle for sirve para recorrer la cadena, desde el primer al último caracter. Luego, dentro del bucle, se imprime cada caracter separado del signo "-". El if que hay dentro del bucle for comprueba que el actual no sea el último caracter, porque en ese caso no habría que escribir el signo "-" (se quiere conseguir "h-o-l-a" y si no estuviera el if se obtendría "h-o-l-a-").

5.8. Profundizando Funciones:

5.8.1 Paso de parámetros

Se explicarán algunos detalles adicionales sobre la definición y uso de funciones, para ampliar el artículo de funciones en php.

5.8.1.1 Paso de parámetros

Los parámetros son los datos que reciben las funciones y que utilizan para realizar las operaciones de la función. Una función puede recibir cualquier número de parámetros, incluso ninguno. A la hora de definir la función, en la cabecera, se definen los parámetros que va a recibir.

```
function f1 ($parametro1, $parámetro2)
```

Así se define una función llamada f1 que recibe dos parámetros. Como se puede observar, no se tiene que definir el tipo de datos de cada parámetro.

Los parámetros tienen validez durante la ejecución de la función, es decir, tienen un ámbito local a la función donde se están recibiendo. Cuando la función se termina, los parámetros dejan de existir.

5.8.1.2 Los parámetros se pasan por valor

El paso de parámetros en PHP se realiza por valor. "Por valor" es una manera típica de pasar parámetros en funciones, quiere decir que el cambio de un dato de un parámetro no actualiza el dato de la variable que se pasó a la función. Por ejemplo, cuando se invoca una función pasando una variable como parámetro, a pesar de que se cambie el valor del parámetro dentro de la función, la variable original no se ve afectada por ese cambio. Puede que se vea mejor con un ejemplo:

```
function porvalor ($parametro1){
$parametro1="hola";
echo "<br>" . $parametro1; //imprime "hola"
}

$mivariable = "esto no cambia";
porvalor ($mivariable);
echo "<br>" . $mivariable; //imprime "esto no cambia"
```

Esta página tendrá como resultado:

hola
esto no cambia

5.8.1.3 Paso de parámetros por referencia

En contraposición al paso de parámetros por valor, está el paso de parámetros por referencia. En este último caso, el cambio del valor de un parámetro dentro de una función sí afecta al valor de la variable original.

Se pueden pasar los parámetros por referencia si, en la declaración de la función, se coloca un "&" antes del parámetro.

```
<?
function porreferencia(&$cadena)
{
$cadena = 'Si cambia';
}
$str = 'Esto es una cadena';
porreferencia ($str);
echo $str; // Imprime 'Si cambia'
?>
```

Este script mostrará por pantalla 'Si cambia'.

5.8.1.4 Parámetros por defecto

Pueden definirse valores por defecto para los parámetros. Los valores por defecto sirven para que los parámetros contengan un dato predefinido, con el que se inicializarán si no se le pasa ningún valor en la llamada de la función. Los valores por defecto se definen asignando un dato

al parámetro al declararlo en la función.

```
function pordefecto ($parametro1="pepe";$parametro2=3)
```

Para la definición de función anterior, $parametro1 tiene como valor por defecto "pepe", mientras que $parametro2 tiene 3 como valor por defecto.

Si se le llama a la función sin indicar valores a los parámetros, estos tomarán los valores asignados por defecto:

```
pordefecto () // $parametro1 vale "pepe" y $parametro2 vale 3
```

Si se le llama a la función indicando un valor, este será tenido en cuenta para el primer parámetro.

```
pordefecto ("hola") // $parametro1 vale "hola" y $parametro2 vale 3
```

Nota: es obligatorio a declarar todos los parámetros con valores por defecto al final.

5.9 Retorno de valores

Las funciones pueden retornar valores. Para ello se utiliza la palabra "return" indicando a continuación el dato o variable que tienen que retornar. La función puede tener múltiples return, aunque sólo devolverá datos por uno de ellos cada vez porque, cuando se llama a return, se termina la ejecución de la función devolviendo el dato indicado.

5.9.1 Ejemplo de función IVA

En esta oportunidad se procederá a ver un nuevo ejemplo para ilustrar el funcionamiento de una función un poco más avanzada, que utiliza parte de los nuevos conceptos introducidos en este artículo.

Se trata de hacer una función que calcula el IVA y que recibe dos parámetros. Uno el valor sobre el que se calcula y el otro el porcentaje a aplicar. Si no se indica el porcentaje de IVA se entiende que es el nueve por ciento (9%).

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

<body>
<?
function iva($base,$porcentaje=9){
  return $base * $porcentaje /100;
}

echo iva(1000) . "<br>";
echo iva(1000,7) . "<br>";
echo iva(10,0) . "<br>";
?>

</body>
</html>
```

Si se han entendido bien los conceptos, este ejemplo no puede resultar difícil. La función

recibe un parámetro llamado $porcentaje con 9 como valor por defecto. Devuelve el porcentaje dado aplicado a la base también indicada por parámetro.

Así pues, en la primera ejecución de la función, como no se indica el porcentaje, se mostrará el 9% de 1000. En la segunda, se muestra el 7% de mil y en la tercera, el 0% de 10.

5.9.2 Retornar múltiples valores

Una función devuelve un único valor. Si queremos hacer que se puedan devolver varios valores distintos tenemos que recurrir a un truco que consiste en devolver un array.

```
function small_numbers()
{
return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

list() se usa para asignar una lista de variables en una sola operación. Después de esa operación, $zero valdrá 0, $one valdrá 1 y $two valdrá 2.

5.10 Control del flujo en PHP: Condiciones IF

La programación exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución.

Como ejemplo, podría hacerse alusión a un script que ejecute una secuencia diferente en función del día de la semana actual.

Este tipo de acciones pueden ser llevadas a cabo gracias a una gama de instrucciones presentes en la mayoría de los lenguajes. En este capítulo se describirán superficialmente algunas de ellas propuestas por PHP y que resultan de evidente utilidad.

Para evitar el complicar el texto, el apartado se limitará a introducir las más importantes dejando de lado otras cuantas que podrán ser fácilmente asimilables a partir de ejemplos prácticos.

5.10.1 Las condiciones if

Cuando se requiere que el programa, llegado a un cierto punto, tome un camino concreto en determinados casos y otro diferente si las condiciones de ejecución difieren, es apropiado servirse del conjunto de instrucciones *if, else* y *elseif*. La estructura de base de este tipo de instrucciones es la siguiente:

```
if (condición)
{
  Instrucción 1;
  Instrucción 2;
  ...
}
else
{
  Instrucción A;
  Instrucción B;
  ...
```

```
}
```

Llegándose a este punto, el programa verificará el cumplimiento o no de la condición. Si la condición es cierta las instrucciones 1 y 2 serán ejecutadas. De lo contrario (*else*), las instrucciones A y B serán llevadas a cabo.

Esta estructura de base puede complicarse un poco más si se toma en cuenta que no necesariamente todo es blanco o negro y que muchas posibilidades pueden darse. Es por ello que otras condiciones pueden plantearse dentro de la condición principal. Se habla por lo tanto de condiciones anidadas que tendrían una estructura del siguiente tipo:

```
if (condición1)
{
  Instrucción 1;
  Instrucción 2;
  ...
}
else
{
  if (condición2)
  {
    Instrucción A;
    Instrucción B;
    ...
  }
  else
  {
    Instrucción X
    ...
  }
}
```

De este modo se podrían introducir tantas condiciones como se requierans dentro de una condición principal.

De gran ayuda es la instrucción *elseif* que permite en una sola línea introducir una condición adicional. Este tipo de instrucción simpifica ligeramente la sintaxis que se acaba de ver:

```
if (condición1)
{
  Instrucción 1;
  Instrucción 2;
  ...
}
elseif (condición2)
{
  Instrucción A;
```

```
  Instrucción B;
  ...
}
else
{
  Instrucción X
  ...
}
```

El uso de esta herramienta resultará claro con un poco de práctica. Aplíquese un ejemplo sencillo de utilización de condiciones. El siguiente programa permitiría detectar la lengua empleada por el navegador y visualizar un mensaje en dicha lengua.

```
<HTML>
<HEAD>
<TITLE>Detector de Lengua</TITLE>
</HEAD>
<BODY>
<?
//Antes de nada se introducen mensajes en forma de variables
$espanol="Hola";
$ingles="Hello";
$frances="Bonjour";

//Ahora se lee del navegador cuál es su lengua oficial
$idioma=substr($HTTP_ACCEPT_LANGUAGE,0,2);

//Formulamos las posibilidades que se pueden dar
if ($idioma == "es")
{echo "$espanol";}
elseif ($idioma=="fr")
{echo "$frances";}
else
{echo "$ingles";}
?>
</BODY>
</HTML>
```

Para poder ver el funcionamiento de este script es necesario cambiar el idioma preferido lo cual puede ser realizado a partir del menú de opciones del navegador.

Para leer la lengua aceptada por el navegador lo que hacemos es definir una variable (*$idioma*) y, mediante la función *substr*, recogemos las dos primeras letras del código correspondiente al idioma aceptado por el navegador ($HTTP_ACCEPT_LANGUAGE).

La tercera parte del script se encarga de ver si el navegador está en español (es), francés (fr) o en cualquier otro idioma que no sea ninguno de estos dos y de imprimir el mensaje que proceda en cada caso.

A notar que, cuando se trata de comparar variables, se tipea un doble igual "==" en lugar de uno simple "=". Este último queda reservado exclusivamente para asignar valores a variables.

5.11 Control del flujo en PHP: Bucles I

Los ordenadores, como cualquier máquina, están diseñados para realizar tareas repetitivas. Es por ello que los programas pueden aprovecharse de este principio para realizar una determinada secuencia de instrucciones un cierto número de veces. Para ello, se utilizan las estructuras llamadas en bucle, las cuales ayudan, usando unas pocas líneas, a realizar una tarea incluida dentro del bucle un cierto numero de veces definido por nosotros mismos.

PHP propone varios tipos de bucle cada uno con características especificas:

5.11.1 Bucle while

Sin duda el bucle más utilizado y el más sencillo. Se usa para ejecutar las instrucciones contenidas en su interior siempre y cuando la condición definida sea verdadera. La estructura sintáctica es la siguiente.

```
while (condición)
{
  instruccion1;
  instruccion2;
  ...
}
```

Un ejemplo sencillo es este bucle que aumenta el tamaño de la fuente en una unidad a cada nueva vuelta por el bucle:

```
<?
$size=1;
While ($size<=6)
{
  echo"<font size=$size>Tamaño $size</font><br>\n";
  $size++;
}
?>
```

A modo de explicación, se dirá que, antes que nada, ha de definirse el valor de la variable que va a evaluarse en la condición. Algo absolutamente obvio pero fácil de olvidar. En este caso se le ha atribuido un valor de 1 que corresponde a la letra más pequeña.

El paso siguiente es crear el bucle en el que se impone la condición que la variable no exceda el valor de 6.

La instrucción a ejecutar será imprimir en el documento un código HTML en el que la etiqueta *font* y el mensaje que contiene varían a medida que *$size* cambia su valor.

El siguiente paso es incrementar en una unidad el valor de $size. Esto se puede hacer con una expresión como la mostrada en el bucle ($size++) que en realidad es sinónima de:

$size=$size+1. Obsérvense otras de estas abreviaciones más adelante.

5.11.2 Otro ejemplo del buble While

El bucle while se suele utilizar cuando no se sabe exactamente cuantas iteraciones se deben realizar antes de culminar. Se procederá a utilizarlo en otro ejemplo, en el que hay que recorrer una cadena hasta encontrar un carácter dado. Si lo encuentra, escribir su posición. Si no, escribir que no se ha encontrado.

> **Nota:** Para hacer este ejercicio es necesario conocer la función de cadena strlen(), que obtiene la longitud de la cadena que se le pase por parámetro.
>
> int strlen (string cad)
> Devuelve un entero igual a la longitud de la cadena.

```
<?
$cadena = "hola a todo el mundo";

//recorro la cadena hasta encontrar una "m"
$i=0;
while ($cadena[$i]!="m" && $i< strlen($cadena)){
   $i++;
}

if ($i==strlen($cadena))
   echo "No se encuentra...";
else
   echo "Está en la posición $i";
?>
```

En este ejemplo se define una cadena con el valor "hola a todo el mundo". Posteriormente se recorre esa cadena hasta el final de la cadena o hasta encontrar el caracter "m", utilizando una variable $i que lleva la cuenta de los caracteres recorridos.

Al final del bucle while, si se salió porque se encontró el caracter "m", la variable $i valdrá un número menor que la longitud de la cadema. Si se salió por llegar al final de la cadena, la variable $i valdrá lo mismo que la longitud en caracteres de esa cadena. En el condicional simplemente se comprueba si $i vale o no lo mismo que la longitud de la cadena, mostrando los mensajes adecuados en cada caso.

5.11.3 Bucle do/while

Este tipo de bucle no difiere en exceso del anterior. La sintaxis es la siguiente:

```
do
{
  instruccion1;
  instruccion2;
  ...
}
while (condición)
```

La diferencia con respecto a los bucles *while* es que este tipo de bucle evalúa la condición al final con lo que, incluso siendo falsa desde el principio, éste se ejecuta al menos una vez.

Control del flujo en PHP: Bucles II

5.11.4 Bucle for

PHP está provisto de otros tipos de bucle que también resultan muy prácticos en determinadas situaciones. El más popular de ellos es el bucle *for* que, como para los casos anteriores, se

encarga de ejecutar las instrucciones entre llaves. La diferencia con los anteriores radica en cómo se plantea la condición de finalización del bucle. Para aclarar su funcionamiento se expresará el ejemplo de bucle *while* visto en el capítulo anterior en forma de bucle *for*:

```
<?
For ($size=1;$size<=6;$size++)
{
  echo"<font size=$size>Tamaño $size</font><br>\n";
}
?>
```

Las expresiones dentro del paréntesis definen respectivamente:

-Inicialización de la variable. Valida para la primera vuelta del bucle.

-Condición de evaluación a cada vuelta. Si es cierta, el bucle continua.

-Acción a realizar al final de cada vuelta de bucle.

5.11.5 Bucle foreach

Este bucle, implementado en las versiones de PHP4, ayuda a recorrer los valores de un array lo cual puede resultar muy útil por ejemplo para efectuar una lectura rápida del mismo. Téngase nuevamente en cuenta que un array es una variable que guarda un conjunto de elementos (valores) catalogados por claves.

La estructura general es la siguiente:

```
Foreach ($array as $clave=>$valor)
{
  instruccion1;
  instruccion2;
  …;
}
```

Un ejemplo práctico es la lectura de un array lo cual podría hacerse del siguiente modo:

```
<?
$moneda=array("España"=> "Peseta","Francia" => "Franco","USA" => "Dolar");
Foreach ($moneda as $clave=>$valor)
{
  echo "Pais: $clave Moneda: $valor<br>";
}
?>
```

Este script se encargaría de mostrar por pantalla el contenido del array *$moneda*. No resultaría mala idea crear una función propia basada en este bucle que permitiese visualizar arrays

mono-dimensionales y almacenarla en una librería. Esta función podría ser definida de esta forma:

```
Function mostrar_array ($array)
{
Foreach ($array as $clave=>$valor)
{echo "$clave=>$valor<br>";}
}
```

5.11.6 Break y continue

Estas dos instrucciones se introducen dentro de la estructura y nos sirven respectivamente para escapar del bucle o saltar a la iteración siguiente. Pueden resultarnos muy prácticas en algunas situaciones.

6. Operadores

Las variables, como base de información de un lenguaje, pueden ser creadas, modificadas y comparadas con otras por medio de los llamados operadores. En los capítulos anteriores se han utilizado en los ejemplos algunos de ellos.

En este capítulo se pretende listar los más importantes y así dar constancia de ellos para futuros ejemplos.

6.1 Operadores aritméticos

Nos permiten realizar operaciones numéricas con nuestras variables

+	Suma
-	Resta
*	Multiplicación
/	División
%	Devuelve el resto de la división

Referencia: El operador aritmético que puede resultar más desconocido para los lectores es el operador %.Se explica con mayor detenimiento su funcionamiento y un ejemplo en el que es útil en el apéndice: Listas de elementos con colores alternos en PHP.

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

==	Igualdad
!=	Desigual
<	Menor que
<=	Menor igual que
>	Mayor que
>=	Mayor igual que

6.3 Operadores lógicos
Se usan en combinación con los operadores de comparación cuando la expresión de la condición lo requiere.

And	Y
Or	O
!	No

6.4 Operadores de incremento
Sirven para aumentar o disminuir de una unidad el valor de una variable

++$variable	Aumenta de 1 el valor de $variable
--$variable	Reduce de uno el valor de $variable

6.5 Operadores combinados
Una forma habitual de modificar el valor de las variables es mediante los operadores combinados:

$variable **+=** 10	Suma 10 a $variable
$variable **-=** 10	Resta 10 a $variable
$variable **.=** "añado"	Concatena las cadenas $variable y "añado"

Este tipo de expresiones no son más que abreviaciones de otras formas más clásicas:

$variable += 10

es lo mismo que:

$variable = $variable+10

6.6 Pasar variables por la URL

Los bucles y condiciones son muy útiles para procesar los datos dentro de un mismo script. Sin embargo, en un sitio Internet, las páginas vistas y los scripts utilizados son numerosos. Muy a menudo se necesita que los distintos scripts estén conectados unos con otros y que se sirvan de variables comunes. Por otro lado, el usuario interacciona por medio de formularios cuyos campos han de ser procesados para poder dar una respuesta. Todo este tipo de factores dinámicos han de ser eficazmente regulados por un lenguaje como PHP.

Es posible que en este punto el lector percate que las variables de un script tienen una validez exclusiva para el script y que resulta imposible conservar su valor cuando se ejecuta otro archivo distinto aunque ambos estén enlazados. Existen varias formas de enviar las variables de una página a otra de manera a que la página destino reconozca el valor asignado por el script de origen:

6.6.1 Pasar variables por URL

Para pasar las variables de una página a otra puede introducirse dicha variable dentro del enlace hipertexto de la página destino. La sintaxis sería la siguiente:

```
<a href="destino.php?variable1=valor1&variable2=valor2&...">Mi enlace</a>
```

Puede observarse que estas variables no poseen el símbolo $ delante. Esto es debido a que en realidad este modo de pasar variables no es específico de PHP sino que es utilizado por otros lenguajes.

Ahora lavariable pertenece también al entorno de la página *destino.php* y está lista para su explotación.

> **Nota:** No siempre se definen automáticamente las variables recibidas por parámetro en las páginas web, depende de una variable de configuración de PHP: register_globals, que tiene que estar activada para que así sea.

Para aclarar posibles dudas, obsérvese lo siguiente en forma de ejemplo. Se tendrán pues dos páginas, *origen.html* (no es necesario darle extensión PHP puesto que no hay ningún tipo de código) y *destino.php*:

```
<HTML>
<HEAD>
<TITLE>origen.html</TITLE>
</HEAD>
```

```
<BODY>
<a href="destino.php?saludo=hola&texto=Esto es una variable texto">Paso variables saludo y texto a la página destino.php</a>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>destino.php</TITLE>
</HEAD>
<BODY>
<?
echo "Variable \$saludo: $saludo <br>\n";
echo "Variable \$texto: $texto <br>\n"
?>
</BODY>
</HTML>
```

6.6.2 $HTTP_GET_VARS

Recuérdese que es posible recopilar en una variable tipo array el conjunto de variables que han sido enviadas al script por este método a partir de la variable de sistema $HTTP_GET_VARS, que es un array asociativo. Utilizándolo quedaría así:

```
<?
echo "Variable \$saludo: $HTTP_GET_VARS["saludo"] <br>\n";
echo "Variable \$texto: $HTTP_GET_VARS["texto"] <br>\n"
?>
```

> **Nota:** Aunque se puedan recoger variables con este array asociativo o utilizar directamente las variables que se definen en nuestra página, resulta más seguro utilizar $HTTP_GET_VARS por dos razones, la primera que así se tiene seguridad que esa varible viene realmente de la URL y la segunda, que así el código será más claro cuando lo volvamos a leer, porque quedará especificado que esa variable estamos recibiéndola por la URL.

6.6.3 $_GET

A partir de la versión 4.1.0 de PHP se ha introducido el array asociativo $_GET, que es idéntico a $HTTP_GET_VARS, aunque un poco más corto de escribir.

6.7 Procesar variables de formularios

Este tipo de transferencia es de gran utilidad ya que permite la interacción directamente con el usuario.

El proceso es similar al explicado para las URLs. Primeramente, se presenta una página inicial con el formulario clásico a rellenar y las variables son recogidas en una segunda página que las procesa:

Nota: No siempre se definen automáticamente las variables recibidas por el formulario en las páginas web, depende de una variable de configuración de PHP: register_globals, que tiene que estar activada para que así sea. Ver comentarios del artículo al final de la página para más información.

```
<HTML>
<HEAD>
<TITLE>formulario.html</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="destino2.php">
Nombre<br>
<INPUT TYPE="TEXT" NAME="nombre"><br>
Apellidos<br>
<INPUT TYPE="TEXT" NAME="apellidos"><br>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>destino2.php</TITLE>
</HEAD>
<BODY>
<?
echo "Variable \$nombre: $nombre <br>\n";
echo "Variable \$apellidos: $apellidos <br>\n"
?>
</BODY>
</HTML>
```

6.7.1 $HTTP_POST_VARS

Téngase en cuenta que es posible recopilar en una variable tipo array el conjunto de variables que han sido enviadas al script por este método a partir de la variable de sistema $HTTP_POST_VARS.

echo "Variable \$nombre: " . $HTTP_POST_VARS["nombre"] . "
\n";

Nota: Aunque sea posible recoger variables con este array asociativo o utilizar directamente las variables que se definen en una página, resulta más seguro utilizar $HTTP_POST_VARS por dos razones, la primera que así se tiene seguridad que esa variable viene realmente de un formulario y la segunda, que así el código será más claro cuando lo vuelva a leerse, porque quedará especificado que esa variable esta recibiéndose por un formulario.

6.7.2 $_POST

A partir de PHP 4.1.0 se pueden recoger las variables de formulario utilizando también el array asociativo $_POST, que es el mismo que $HTTP_POST_VARS, pero más corto de escribir.

6.7.3 Ejemplo de restricción de acceso por edad

Para continuar aportando ejemplos al uso de formularios vamos a realizar una página que muestra solicita la edad del visitante y, dependiendo de dicha edad, permita o no visualizar el contenido de la web. A los mayores de 18 años se les permite ver la página y a los menores no.

El ejemplo es muy sencillo y no valdría tal cual está para utilizarlo a modo de una verdadera restricción de acceso. Únicamente nos sirve para saber cómo obtener datos de un formulario y como tratarlos para realizar una u otra acción, dependiendo de su valor.

La página del formulario, que ha sido llamado edad.php tendría esta forma:

```
<html>
<head>
  <title>Restringir por edad</title>
</head>

<body>

<form action="edad2.php" method="post">
Escribe tu edad: <input type="text" name="edad" size="2">
<input type="submit" value="Entrar">
</form>

</body>
</html>
```

Esta es una página sin ningún código PHP, simplemente tiene un formulario. Fíjese atención en el action del formulario, que está dirigido hacia una página llamada edad2.php, que es la que recibirá el dato de la edad y mostrará un contenido u otro dependiendo de ese valor. Su código es el siguiente:

```
<html>
<head>
   <title>Restringir por edad</title>
</head>

<body>

<?
$edad = $_POST["edad"];
echo "Tu edad: $edad<p>";

if ($edad < 18) {
   echo "No puedes entrar";
}else{
   echo "Bienvenido";
}
?>
</body>
</html>
```

Este último código tampoco no debe resultar extraño en lo absoluto. Simplemente se recibe la edad, utilizando el array $_POST. Luego se muestra la edad y se ejecuta una expresión condicional en función de que la edad sea menor que 18. En caso positivo (edad menor que 18), se muestra un mensaje que informa de que no se deja acceder al página. En caso negativo (mayor o igual a 18) se muestra un mensaje de bienvenida.

6.8. Autollamada de páginas

Al incluir un formulario en una página se debe indicar, a través del atributo action, el nombre del archivo PHP al que se enviarán los datos escritos en el formulario. De este modo, para un esquema de envío de datos por formulario, pueden participar dos páginas: una que contiene el formulario y otra que recibe los datos de dicho formulario.

Lo mismo ocurre cuando se envian variables por una URL. Se tiene una página que contendrá el enlace y otra página que recibirá y tratará esos datos para mostrar unos resultados.

En el presente artículo se verá cómo se pueden enviar y recibir datos de un formulario con una única página. Asimismo, cómo en la misma página se pueden tener enlaces con paso de variables por URL y además, pueden recogerse y tratarse esos datos con la misma página. A este efecto puede llamarse "autollamada de páginas", también se le suele llamar como "Formularios reentrantes" o términos similares. Es muy interesante conocer el modo de funcionamiento de estos scripts, porque serán muy habituales en nuestras páginas PHP y ayudan mucho a tener los códigos ordenados.

En ambos casos, para formularios o envío de datos por la URL, se debe seguir un esquema como este:

- Comprobar si recibo datos por URL o por formulario
- Si no recibo datos
- Muestro el formulario o los enlaces que pasan variables.
- Si recibo datos
- Entonces tengo que procesar el formulario o las variables de la URL

6.8.1 Para un formulario

Véase a continuación como sería el código de un formulario reentrante.

```
<html>
<head>
   <title>Me llamo a mi mismo...</title>
</head>

<body>
<?
if (!$_POST){
?>
   <form action="auto-llamada.php" method="post">
   Nombre: <input type="text" name="nombre" size="30">
   <br>
   Empresa: <input type="text" name="empresa" size="30">
   <br>
   Telefono: <input type="text" name="telefono" size=14 value="+34 " >
   <br>
   <input type="submit" value="Enviar">
   </form>
<?
}else{
   echo "<br>Su nombre: " . $_POST["nombre"];
   echo "<br>Su empresa: " . $_POST["empresa"];
   echo "<br>Su Teléfono: " . $_POST["telefono"];
}
?>
</body>
</html>
```

En el ejemplo, el primer paso es conocer si se están recibiendo o no datos por un formulario. Para ello se comprueba con un enunciado if si existe o no una variable $_POST.

En concreto if (!$_POST) querría decir algo como "Si no existen datos venidos de un formulario". En caso de que no existan, muestro el formulario. En caso de que sí existan,

recojo los datos y los imprimo en la página.

6.8.2 Para paso de variables por URL

La idea es la misma. Comprobar con un enunciado if si se reciben o no datos desde una URL. Veamos el código a continuación. Se trata de una página que muestra una serie de enlaces para ver las tablas de multiplicar de el 1 hasta el 10. Cada uno de los enlaces muestra una tabla de multiplicar. Pulsando el primer enlace podemos ver la tabla del 1, pulsando el segundo la tabla del 2, etc.

Recuérdese que la página se llama a si misma. Para comprenderla más fácilmente será interesante.

```
<html>
<head>    <title>Tablas de multiplicar</title>
</head>

<body>
<?
if (!$_GET){
   for ($i=1;$i<=10;$i++){
      echo "<br><a href='ver_tabla.php?tabla=$i'>Ver la tabla del $i</a>\n";
   }
} else {
   $tabla=$_GET["tabla"];
?>
   <table align=center border=1 cellpadding="1">
<?
   for ($i=0;$i<=10;$i++){
      echo "<tr><td>$tabla X $i</td><td>=</td><td>" . $tabla * $i . "</td></tr>\n";
   }
?>

   </table>
<?
}
?>
</body>
</html>
```

Este código es un poco más complicado, porque hace un poco más de cosas que el anterior, pero para el asunto que nos ocupa que es la autollamada de páginas, todo sigue igual de simple.

Hay que fijarse en el if que comprueba si se reciben o no datos por URL: if (!$_GET), que querría decir algo como "Si no se reciben variables por la URL".

En caso positivo (no se reciben datos por URL) se muestran los enlaces para ver cada una de las tablas y en caso de que sí se reciban datos, se muestra la tabla de multiplicar del número que se está recibiendo en la URL.

Para hacer mostrar los enlaces y las tablas de multiplicar se utilizan bucles for, se amplía sobre bucles for en el apartado siguiente.

6.9. Utilización de las cookies

Sin duda este término resultará familiar para muchos. Algunos lo habrán leído u oído pero no saben de qué se trata. Otros sin embargo sabrán que las cookies son unas informaciones almacenadas por un sitio web en el disco duro del usuario. Esta información es almacenada en un archivo tipo texto que se guarda cuando el navegador accesa al sitio web.

En los viajes por la Red suelen visitarse gran cantidad de páginas, muchas de ellas bastante complicadas que implementan distintos servicios de Internet. Estas páginas tienen que guardar

distintas informaciones acerca de un usuario, por ejemplo su nombre, su edad o su color preferido. Para ello cuentan con una serie de mecanismos en el servidor como bases de datos u otro tipo de contenedores, pero hay un mecanismo mucho más interesante de guardar esa información que los propios recursos del servidor, que es el propio ordenador del usuario.

En los ordenadores del cibernauta se guardan muchos datos que necesitan conocer las páginas web cada vez que se entra en la página, estas pequeñas informaciones son las cookies: estados de variables que se conservan de una visita a otra en el ordenador del cliente.

Como es un poco peligroso que las páginas web a las que se tiene acceso se dediquen a introducir cosas en el ordenador del visitante, las cookies están altamente restringidas. Para empezar, solamente pueden guardarse en ellas textos, nunca programas, imágenes, etc. Además, los textos nunca podrán ocupar más de 1 K, de modo que nadie podría inundar el ordenador del visitante a base de cookies. Estas restricciones, unidas a la necesidad de **poner una fecha de caducidad** a las cookies para que estas se guarden, hacen que el aceptar cookies no signifique un verdadero problema para la integridad de los sistemas.

6.9.1 Ejemplos de uso de las cookies

Un ejemplo típico de las cookies podría ser un contador de las veces que un usuario tiene acceso a una página. Podría ponerse una cookie en el ordenador del cliente donde se tendría una variable que lleva la cuenta de las veces que ha tenido acceso a la página y cada vez que eso sucede, la variable se incrementa en uno.

Otro ejemplo típico es el que guarda el perfil del usuario. Si el usuario tiene acceso a contenidos determinados puede enviarsele una cookie que le marca como interesado en un tema. A medida que va entrando a distintos sitios se le va encasillando como joven, adulto, hombre, mujer, o lo que proceda. Conociendo el perfil de un usuario se le pueden ofrecer un tipo de productos o servicios orientados a sus gustos o necesidades.

6.9.2 Cómo se utilizan las cookies

Para trabajar con cookies tenemos que utilizar un lenguaje de programación avanzado como Javascript o ASP, PHP, etc. No podemos entonces trabajar con cookies si solamente nos dedicamos a utilizar el HTML, que ya sabemos que es un poco limitado para cosas que se salgan de mostrar contenido en páginas estáticas.

6.9.3 Polémica de las cookies

Existe un problema con las cookies y es que nos cortan hasta cierto punto la privacidad. Lo que se señalaba anteriormente acerca de guardar el perfil de un usuario puede llegar a ser un problema para el mismo, por que podría decirse que éste está siendo vigilado y apuntado cada uno de sus movimientos, lo que puede convertirse en un abuso de información que no tiene por qué pertenecer a nadie más que al cibernauta. Las empresas que más utilizan esta clasificación del personal son los AD-Servers (servidores de banners) como el de Doubleclick. No se desea entrar aquí más en esta polémica, pero si despertar la inquietud de que posiblemente son elementos potencialmente nocivos a la privacidad.

Es posible, por supuesto, ver estos archivos. Para abrirlos hay que ir al directorio C:\Windows\Cookies para los usuarios de Internet Explorer a partir de versión 4 o a C:\...\Netscape\Users\defaultuser para usuarios de Netscape. Como podrá comprobarse, en la mayoría de los casos la información que se puede obtener es indescifrable.

La utilidad principal de las cookies es la de poder identificar al navegador una vez éste visita el sitio por segunda vez y así, en función del perfil del cliente dado en su primera visita, el sitio puede adaptarse dinámicamente a sus preferencias (lengua utilizada, colores de pantalla, formularios rellenados total o parcialmente, redirección a determinadas páginas...).

Crear un archivo cookies, modificar o generar una nueva cookie es algo que puede hacerse a partir de la función SetCookie:

```
setcookie("nombre_de_la_cookie",valor,expiracion);
```

Aplíquese un ejemplo sencillo. Imagínese que se quiere introducir en una variable cookie el nombre del visitante. El nombre ha podido ser previamente recogido por un formulario tal y como se ha visto:

```
setcookie("persona",$nombre,time()+86400*365);
```

De este modo ha sido creada una cookie llamada persona que tiene como valor el contenido de la variable $nombre y tendrá una duración de 1 año a partir de su creación (el tiempo time() actual en segundos sumado a un año en segundos).

Es importante que la creación de la cookie sea previa a la apertura del documento HTML. En otras palabras, las llamadas a la función setcookie() deben ser colocadas antes de la etiqueta HTML.

Por otra parte, es interesante señalar que el hecho de que definir una cookie ya existente implica el borrado de la antigua. Del mismo modo, el crear una primera cookie conlleva la generación automática del archivo texto.

Para utilizar el valor de la cookie en un script tan sólo se tendrá que llamar la variable que define la cookie. ¡Realmente sencillo!

Hay que tener cuidado sin embargo de no definir variables en un script con el mismo nombre que las cookies puesto que PHP privilegiará el contenido de la variable local con respecto a la cookie y no dará un mensaje de error. Esto puede conducir a errores realmente difíciles de detectar.

Recuérdese que es posible recopilar en una variable tipo array el conjunto de cookies almacenadas en el disco duro del internauta mediante la variable de servidor $HTTP_COOKIE_VARS

Las cookies son una herramienta fantástica para personalizar nuestra página pero hay que ser cautos ya que, por una parte, no todos los navegadores las aceptan y por otra, se puede deliberadamente impedir al navegador la creación de cookies. Es por ello que resultan un complemento y no una fuente de variables infalible para UN sitio.

7. Sesiones I

En los programas que se han visto hasta ahora, se han utilizado variables que sólo existían en el archivo que era ejecutado. Cuando se cargaba otra página distinta, los valores de estas variables se perdían a menos que se tomase la molestia de pasarlos por la URL o inscribirlos en las cookies o en un formulario para su posterior explotación. Estos métodos, aunque útiles, no son todo lo prácticos que podrían en determinados casos en los que la variable que se desea conservar ha de ser utilizada en varios scripts diferentes y distantes los unos de los otros.

Podría pensarse que ese problema puede quedar resuelto con las cookies ya que se trata de variables que pueden ser invocadas en cualquier momento. El problema, ya se ha dicho, es que las cookies no son aceptadas ni por la totalidad de los usuarios ni por la totalidad de los

navegadores, lo cual implica que una aplicación que se sirviera de las cookies para pasar variables de un archivo a otro no sería 100% infalible. Es importante a veces pensar en "la inmensa minoría", sobre todo en aplicaciones de comercio electrónico donde debe captarse la mayor cantidad de clientes posibles y donde los scripts deben estar preparados ante cualquier eventual deficiencia del navegador del cliente.

Resulta pues necesaria la capacidad de declarar ciertas variables que puedan ser reutilizadas tantas veces como se requiera o desee dentro de una misma sesión. Asúmase, por ejemplo un sitio multilingüe en el que cada vez que se desea imprimir un mensaje en cualquier página se necesite saber en qué idioma debe hacerse. Podría introducirse un script identificador de la lengua del navegador en cada uno de los archivos o bien declarar una variable que fuese valida para toda la sesión y que tuviese como valor el idioma reconocido en un primer momento.

Asúmase también en un carrito de la compra de una tienda virtual donde el cliente va navegando por las páginas del sitio y añadiendo los artículos que quiere comprar a un carrito. Este carrito podría ser perfectamente una variable de tipo array (tabla) que almacena para cada referencia la cantidad de artículos contenidos en el carrito. Esta variable debería ser obviamente conservada continuamente a lo largo de todos los scripts.

Este tipo de situaciones son solventadas a partir de las variables de sesión. Una sesión es considerada como el intervalo de tiempo empleado por un usuario en recorrer una página hasta que abandona el sitio o deja de actuar sobre él durante un tiempo prolongado o bien, sencillamente, cierra el navegador.

PHP permite almacenar variables llamadas de sesión, las cuales, una vez definidas, podrán ser utilizadas durante este lapso de tiempo por cualquiera de los scripts de un sitio. Estas variables serán específicas del usuario de modo que diversas variables sesión del mismo tipo con distintos valores pueden estar coexistiendo para cada una de las sesiones que están teniendo lugar simultáneamente. Estas sesiones tienen además su propio identificador de sesión que será único y específico.

Algunas mejoras referentes al empleo de sesiones han sido introducidas con PHP4. Es a esta previa versión a la que se hará referencia a la hora de explicar las funciones disponibles y la forma de operar. Para los programadores de PHP3 la diferencia mayor es que están obligados a gestionar ellos mismos las sesiones y definir sus propios identificadores de sesión.

Véase en el siguiente apartado la forma de plasmar esta necesidad técnica en los scripts a partir de las funciones que gestionan las sesiones en PHP.

7.1 Sesiones II

Se ha dicho en el apartado anterior que las variables de sesión se diferencian de las variables clásicas en que éstas primeras residen en el servidor, son especificas de un solo usuario definido por un identificador y pueden ser utilizadas en la globalidad de páginas creadas por el programador.

Puede iniciarse una sesión de dos formas distintas:

-Declarando abiertamente la apertura de sesión por medio de la función *session_start()*. Esta función crea una nueva sesión para un nuevo visitante o bien recupera la que está siendo llevada a cabo.

-Declarando una variable de sesión por medio de la función *session_register('variable')*. Esta función, además de crear o recuperar la sesión para la página en la que se incluye también sirve para introducir una nueva variable de tipo sesión.

Las sesiones han de ser iniciadas al principio de nuestro script. Antes de abrir cualquier etiqueta o de imprimir cualquier cosa. En caso contrario recibiremos un error.

Con lo visto, procédase a proponer el ejemplo clásico de utilización de una sesión: un contador. Este contador deberá aumentar de una unidad cada vez que se recarga la página o o se tiene acceso al enlace:

```
<?
session_register('contador');
?>
<HTML>
<HEAD>
<TITLE>contador.php</TITLE>
</HEAD>
<BODY>
<?
If (isset($contador)==0)
{$contador=0;}
++$contador;
echo "<a href=\"contador.php\">Has recargado esta página $contador veces</a>";
?>
</BODY>
</HTML>
```

La condición *if* tiene en cuenta la posibilidad de que la variable *$contador* no haya sido todavía inicializada. La función *isset* se encarga de dar un valor cero cuando una variable no ha sido inicializada.

Otras funciones útiles para la gestión de sesiones son:

Función	Descripción
Session_id()	Nos devuelve el identificador de la sesión
Session_destroy()	Da por abandonada la sesión eliminando variables e identificador.
Session_unregister('variable')	Abandona una variable sesión

7.1.1 Caso de Ejemplo

Apréciese el siguiente sencillo ejemplo de uso de sesiones para ilustrar a los usuarios más inexpertos el manejo de sesiones con PHP en sus procesos más básicos, como son la inicialización de sesiones, comprobar si existen variables de sesión, crearlas y modificarlas.

En este ejemplo se procederá a hacer un sistema para llevar la cuenta de las páginas que ha visto un visitante. Es decir, si entra a una página del sitio por primera vez, se cuenta que ha visto una página, luego con cada página adicional que visite, seguirá acumulándose (1) en la cuenta de páginas vistas dentro del sitio. Del mismo modo, si se recarga una página o se vuelve a la misma página en que ya se estuvo, también se hará que se acumule (1) en su cuenta.

Lo que se tendrá que hacer en este ejercicio es lo siguiente:

- Inicializar la sesión
- Si no se tiene el contador de páginas creado, se inicializa al valor 1
- Si se tiene el contador de páginas creado, se incrementa en 1.

```
<? session_start();
if (!isset($_SESSION["cuenta_paginas"])){
   $_SESSION["cuenta_paginas"] = 1;
}else{
   $_SESSION["cuenta_paginas"]++;
}
?>
<html>
<head>
<title>Contar páginas vistas por un usuario en toda su sesión</title>
</head>

<body>
<?
echo "Desde que entraste has visto " . $_SESSION["cuenta_paginas"] . " páginas";
?>
<br>
<br>
<a href="otracuenta.php">Ver otra página</a>
</body>
</html>
```

Como se ha podido ver, lo primero que se debe hacer es inicializar la sesión con session_start().

Luego de inicializar la sesión se puede trabajar con variables de sesión en cualquier lugar del código PHP. El programador va a crear y utilizar una variable de sesión para llevar la cuenta del número de páginas que se ha visto.

```
$_SESSION["cuenta_paginas"]
```

Para ver si una variable de sesión ha sido creada puede utilizarse la función isset() pasándole como parámetro la variable que desee saberse si ha sido creada. Si no fue creada anteriormente, simplemente la se crea (inicializando su valor a 1). Si fue creada, pues que tiene que incrementarse en (1). Eso se consigue con este código:

```
if (!isset($_SESSION["cuenta_paginas"])){
   $_SESSION["cuenta_paginas"] = 1;
}else{
   $_SESSION["cuenta_paginas"]++;
}
```

Eso es todo! Ahora lo que se supone que se tiene son otras páginas en el mismo sitio que también tienen que incrementar en 1 el contador de páginas vistas por ese usuario en esa sesión.

Eso se podría hacer con el mismo trozo de código, como el que se ha visto antes, colocado al principio del archivo PHP.

```
<? session_start();
if (!isset($_SESSION["cuenta_paginas"])){
   $_SESSION["cuenta_paginas"] = 1;
}else{
   $_SESSION["cuenta_paginas"]++;
}
?>
```

Luego, en cualquier momento, si se desea mostrar el número de páginas vistas en un momento dado, tendríamos que hacer algo como esto:

<? echo "Has visto " . $_SESSION["cuenta_paginas"] . " páginas, contando actualizaciones de paginas"; ?>

8. Trabajar con bases de datos en PHP

Una de las principales ventajas que presenta el trabajar con páginas dinámicas es el poder almacenar los contenidos en bases de datos. De esta forma, el programador puede organizarlos, actualizarlos y buscarlos de una manera mucho más simple.

El lenguaje PHP, ya se ha dicho, ofrece interfaces para el acceso a la mayoría de las bases de datos comerciales y por ODBC a todas las bases de datos posibles en sistemas Microsoft, a partir de las cuales se podrá editar el contenido de un sitio con absoluta sencillez.

Esta interacción se realiza, por un lado, a partir de las funciones que PHP propone para cada tipo de base de datos y, por otro estableciendo un diálogo a partir de un idioma universal: SQL (Structured Query Language) el cual es común a todas las bases de datos. Este lenguaje resulta, como se verá en el módulo de instrucción de SQL, muy potente y fácil de aprender.

En este manual de PHP se limitará el alcance pues a la utilización las instrucciones SQL básicas que serán aprendidas a medida que se explican las diferentes formas de actuar sobre una base de datos dejando para el Módulo de Instrucción de SQL los aspectos más avanzados.

Como base ejemplo de estos capítulos se ha elegido MySQL, sin duda la base de datos más extendida en combinación con PHP. Su gratuidad, eficiencia y simplicidad la han hecho una de las opciones más demandadas en el mercado del desarrollo web.

Ya se ha explicado en apartados anteriores su instalación a la vez que se han presentado los comandos de base que pueden permitir abordarla con una relativa facilidad.

En caso de utilizar cualquier otra base compatible, las correcciones a llevar a cabo con respecto a los ejemplos aquí presentados no son excesivamente grandes y la lectura de esos capítulos sigue siendo de gran utilidad.

Una vez instalado MySQL y antes de poder comenzar con los ejemplos, será necesario llevar a cabo las siguientes operaciones:

-Dentro de MySQL, crear la base de datos ejemplo con la siguiente sentencia:

```
create database ejemplo;
```

-Seleccionar la base ejemplo como la base a utilizar:

```
use ejemplo
```

-Crear a continuación la tabla clientes a partir de la siguiente sentencia:

```
create table clientes (
nombre varchar(100),
telefono varchar(100)
```

```
);
```

Ahora ya se dispone de una tabla vacía. Sólo queda comenzar a llenarla con los datos que se irán insertando.

8.1 Introducción de nuevos registros

Una vez creada la tabla *clientes* en la base de datos *ejemplo*, el paso siguiente sea llenarla con registros.

Los datos del registro pueden ser recogidos, por ejemplo, a partir de un formulario. Aquí se propone un simple documento HTML que recoge los datos y los envía a una página PHP que se encarga de procesarlos:

```
<HTML>
<HEAD>
<TITLE>Insertar.html</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Insertar un registro</h1>
<br>
<FORM METHOD="POST" ACTION="insertar.php">
Nombre<br>
<INPUT TYPE="TEXT" NAME="nombre"><br>
Teléfono<br>
<INPUT TYPE="TEXT" NAME="telefono"><br>
<INPUT TYPE="SUBMIT" value="Insertar">
</FORM>
</div>
</BODY>
</HTML>
```

Llegados a la página destino del formulario (*insertar.php*), lo primero que habrá que hacer es establecer un vínculo entre el programa y la base de datos. Esta conexión se lleva a cabo con la función *mysql_connect*. A continuación, deberá generarse una orden de inserción del registro en lenguaje SQL. Esta orden será ejecutada por medio de la función *mysql_db_query*.

En esta función se especificará primeramente la base de datos sobre la que se quiere actuar y a continuación introducir la sentencia SQL:

```
<HTML>
<HEAD>
<TITLE>Insertar.php</TITLE>
</HEAD>
```

```
<BODY>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");
//Ejecucion de la sentencia SQL
mysql_db_query("ejemplo","insert into clientes (nombre,telefono) values
('$nombre','$telefono')");
?>
<h1><div align="center">Registro Insertado</div></h1>
<div align="center"><a href="lectura.php">Visualizar el contenido de la base</a></div>
</BODY>
</HTML>
```

Los parámetros user y password son definidos por el creador de la base de datos. Es conveniente en un principio, al crear las bases de datos, trabajar sin ellos con lo cual se dejará las cadenas correspondientes vacias: "".

Además de la propia inserción, el programa avisa de la introducción del registro y ofrece un enlace hacia una página de lectura la cual será comentada a continuación.

No se ahondará en la descripción de la orden SQL, en cuanto a comprensión acerca de cómo introducir registros, se dedicarán apartados y prácticas en el módulo de instrucción de SQL.

8.2 Selección y lectura de registros

Dentro de una base de datos, organizada por tablas, la selección de una tabla entera o de un cierto numero de registros resulta una operación rutinaria.

Aquí se ilustra una forma bastante clásica de mostrar en pantalla a partir de un bucle los registros seleccionados por una sentencia SQL:

```
<HTML>
<HEAD>
<TITLE>lectura.php</TITLE>
</HEAD>
<BODY>
<h1><div align="center">Lectura de la tabla</div></h1>
<br>
<br>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//Ejecutamos la sentencia SQL
$result=mysql_db_query("ejemplo","select * from clientes");
?>
<table align="center">
<tr>
<th>Nombre</th>
<th>Teléfono</th>
</tr>
```

```
<?
//Mostramos los registros
while ($row=mysql_fetch_array($result))
{
echo '<tr><td>'.$row["nombre"].'</td>';
echo '<td>'.$row["telefono"].'</td></tr>';
}
mysql_free_result($result)
?>
</table>

<div align="center">
<a href="insertar.html">Añadir un nuevo registro</a><br>
<a href="actualizar1.php">Actualizar un registro existente</a><br>
<a href="borrar1.php">Borrar un registro</a><br>
</div>

</BODY>
</HTML>
```

Los pasos a realizar son, en un principio, los vistos para la inserción de un registro: Conexión a la base y ejecución de la sentencia. Esta vez, la información de dicha ejecución será almacenada en una variable (*$result*).

El siguiente paso será plasmar en pantalla la información recogida en $result. Esto lo haremos mediante la función *mysql_fetch_array* que devuelve una variable array con los contenidos de un registro a la vez que se posiciona sobre el siguiente. El bucle *while* nos permite leer e imprimir secuencialmente cada uno de los registros..

La función *mysql_free_result* se encarga de liberar la memoria utilizada para llevar a cabo la consulta. Aunque no es necesaria su utilización, resulta altamente aconsejable.

8.3 Actualizacion de un registro

Para mostrar cómo se actualiza un registro presente en nuestra base de datos, se ejecutará el procedimiento a partir de un caso un poco más complejo para de tal manera empezar a familiarizar al lector con estas operaciones. Se realizarán un par de scripts que permitan cambiar los números de teléfono de las distintas personas presentes en la base de datos. El nombre de estas personas, así como el nuevo numero de teléfono, serán recogidos por medio de un formulario.

El archivo del formulario va a ser esta vez un script PHP en el que se efectuará una llamada a la base de datos para construir un menú desplegable donde aparezcan todos los nombres. El script quedaría así:

```
<HTML>
<HEAD>
<TITLE>Actualizar1.php</TITLE>
</HEAD>
<BODY>
```

```
<div align="center">
<h1>Actualizar un registro</h1>
<br>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

echo '<FORM METHOD="POST" ACTION="actualizar2.php">Nombre<br>';

//Creamos la sentencia SQL y la ejecutamos
$sSQL="Select nombre From clientes Order By nombre";
$result=mysql_db_query("ejemplo",$sSQL);

echo '<select name="nombre">';

//Generamos el menu desplegable
while ($row=mysql_fetch_array($result))
{echo '<option>'.$row["nombre"];}
?>
</select>
<br>
Teléfono<br>
<INPUT TYPE="TEXT" NAME="telefono"><br>
<INPUT TYPE="SUBMIT" value="Actualizar">
</FORM>
</div>

</BODY>
</HTML>
```

La manera de operar para construir el menú desplegable es la misma que para visualizar la tabla. De nuevo se emplea un bucle *while* en combinación con la función *mysql_fetch_array* lo que permite mostrar cada una de las opciones.

El script de actualización será muy parecido al de inserción:

```
<HTML>
<HEAD>
<TITLE>Actualizar2.php</TITLE>
</HEAD>
<BODY>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//Creamos la sentencia SQL y la ejecutamos
$sSQL="Update Clientes Set telefono='$telefono' Where nombre='$nombre'";
mysql_db_query("ejemplo",$sSQL);
?>
```

```
<h1><div align="center">Registro Actualizado</div></h1>
<div align="center"><a href="lectura.php">Visualizar el contenido de la base</a></div>

</BODY>
</HTML>
```

8.4 Borrado de un registro con PHP

Otra de las operaciones elementales que se pueden realizar sobre una base de datos es borrar un registro. Para hacerlo, SQL propone sentencias del tipo *Delete*. Véase con un ejemplo aplicado a la agenda ejemplo. Primero, se creará un menú desplegable dinámico como para el caso de las actualizaciones:

```
<HTML>
<HEAD>
<TITLE>Borrar1.php</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Borrar un registro</h1>
<br>

<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

echo '<FORM METHOD="POST" ACTION="borrar2.php">Nombre<br>';

//Creamos la sentencia SQL y la ejecutamos
$sSQL="Select nombre From clientes Order By nombre";
$result=mysql_db_query("ejemplo",$sSQL);

echo '<select name="nombre">';

//Mostramos los registros en forma de menú desplegable
while ($row=mysql_fetch_array($result))
{echo '<option>'.$row["nombre"];}
mysql_free_result($result)
?>

</select>
<br>
<INPUT TYPE="SUBMIT" value="Borrar">
</FORM>
</div>

</BODY>
</HTML>
```

El siguiente paso es hacer efectiva la operación a partir de la ejecución de la sentencia SQL construida a partir de los datos del formulario:

```
<HTML>
<HEAD>
<TITLE>Borrar2.php</TITLE>
</HEAD>
<BODY>
<?
//Conexion con la base
mysql_connect("localhost","tu_user","tu_password");

//Creamos la sentencia SQL y la ejecutamos
$sSQL="Delete From Clientes Where nombre='$nombre'";
mysql_db_query("ejemplo",$sSQL);
?>

<h1><div align="center">Registro Borrado</div></h1>
<div align="center"><a href="lectura.php">Visualizar el contenido de la base</a></div>

</BODY>
</HTML>
```

9. Subir una aplicación PHP al servidor

9.1. Subir los archivos (file Upload)

Para el proceso de subir archivos, el servidor web debe tener un directorio para la publicación de las páginas web. Ese sería el lugar donde hay que subir los archivos .php.

Dependiendo del proveedor con el que el programador trabaje, el directorio de publicación puede variar. Generalmente, cuando se contrata un alojamiento, se le proporciona al cliente una cuenta de FTP con la cual conectarse al servidor web y transferir los archivos del sitio, además de unos datos para la conexión, que serán el nombre del servidor, el usuario y contraseña para el acceso al FTP.

Al conectarse el programador al servidor con los datos del FTP, que deben ser proporcionados por el proveedor, se tiene acceso a un directorio. Este directorio podría ser el de publicación, aunque generalmente no es así, sino que suele ser un subdirectorio llamado "HTML" o "docs" o algo similar, que depende del directorio de inicio en la conexión FTP. Como decía, este directorio puede tener nombres distintos en proveedores distintos, aunque, en cualquier caso, con una simple pregunta al proveedor se resuelve esa duda.

Los archivos se deben subir al directorio de publicación, o a cualquier subdirectorio de este. En definitiva, tendrán que alojarse en las posibles ubicaciones anteriormente descritas del servidor web y para acceder a ellos bastaría con escribir el nombre del dominio o URL del alojamiento contratado, seguido del nombre del archivo. Si se tuviese un archivo llamado hola.php y el alojamiento se ha contratado para el dominio www.midominio.com, debería

subirse ese archivo al directorio de publicación y se accedería al archivo escribiendo:

http://www.midominio.com/hola.php

Si se crean subdirectorios dentro del directorio de publicación podrá tenerse acceso a ellos escribiendo el nombre del dominio o URL del alojamiento, seguido del nombre del directorio y el nombre del archivo. Por ejemplo, si se crea un subdirectorio llamado páginas y se tiene dentro un archivo llamado pag1.php, podría accederse a él de la siguiente manera:

http://www.midominio.com/paginas/pag1.php

> **Referencia:** hay otro concepto interesante que debería conocerse llegado este punto, y es el "documento por defecto". Éste no es más que el archivo que se envía al navegador si en la URL accedida no se especificaba ningún archivo. Suele llamarse index.html o index.php (o index.asp si el servidor soporta programación en ASP), aunque puede variar de un proveedor a otro. Se hablamos más sobre el apéndice de programación y configuración.

9.1.1 Colocar los archivos PHP fuera del directorio de publicación

Por decir algo más sobre el tema de colocar los archivos, se debe señalar que cualquier cosa que se coloque fuera del directorio de publicación no podrá ser accedida a través del navegador. Es decir, si se crea un directorio que se llame funciones_php en el mismo nivel que el directorio de publicación (fuera del directorio de publicación) no podrá tenerse acceso con el explorador a los archivos que se coloque dentro de ninguna de las maneras. Esto es así porque la URL de inicio del alojamiento corresponde con ese directorio y no puede el usuario moverse hacia debajo de ese directorio con las URLs, que son la manera de especificar al navegador los recursos a los que se quiere acceder.

No sería posible salir del directorio de publicación con una URL como esta, por mucho que se utilice el operador .. (que sirve para acceder al directorio padre).

http://www.midominio.com/../funciones_php/archivo_inalcanzable.php

Sin embargo, colocar algunos contenidos fuera del directorio de publicación puede ser muy útil. Por ejemplo, podrían colocarse allí copias de seguridad de algunos archivos o documentos que simplemente se requieren guardar en el servidor para acceder a ellos desde cualquier parte y con el programa de FTP.

Hay otra utilidad más interesante sobre colocar archivos fuera del directorio de publicación. Se trata de que muchas veces los programadores utilizan en sus programas segmentos de código repetidamente, por ejemplo, para abrir y cerrar bases de datos, para mostrar la cabecera de un portal, para comprobar que un email escrito en un formulario es correcto, etc. Es muy útil separar estos trozos de código en un archivo a parte y llamar a este archivo con las funciones PHP include() o require(). Así, si un día se modifica la cabecera del portal, sólo lo se tendrá que modificar en un archivo, o, si cambia la base de datos que se utiliza sólo se tendría que modificar el archivo que hace la conexión a la base de datos una vez, en lugar de ir cambiándolo en todas las páginas PHP que abrían las bases de datos.

Estos archivos no son páginas independientes, sino segmentos. Seguramente, si se los ejecuta por separado no mostrarían ningún resultado válido, incluso podrían dar mensajes de error.

Por esta razón merece la pena colocarlos en un lugar donde nadie pueda tener acceso: fuera del directorio de publicación. Con PHP si se podrá tener acceso a ese directorio para incluir esos archivos. Solamente deberían utilizarse las funciones PHP include() o require() indicando la ruta para obtener acceso a los archivos.

En el caso de que se tenga una página llamada hola.php en el directorio de publicación y un archivo, que se llama abre_base_datos.php, en el directorio funciones_php, que está fuera del directorio de publicación, si quisiera obtenerse acceso (desde hola.php) al archivo que abre la base de datos se haría así:

include("../funciones_php/abre_base_datos.php")

Téngase siempre en cuenta que desde PHP se puede tener acceso a los archivos que se encuentran fuera del directorio de publicación. Para ello se especifica la ruta adecuada, en la que se utiliza el operador .. para bajar al directorio padre.

Nada más que decir sobre la colocación de los archivos: una vez situado en el directorio de publicación, el programador u operador del servidor web podrá tener acceso a ellos con su navegador y se deberían ejecutar perfectamente. Aunque cabe señalar que, tanto PHP como el servidor donde se trabaje, pueden tener configuraciones distintas y puede que algún detalle de la programación de las páginas no funcione correctamente.

Por ejemplo, el PHP del programador puede declarar o no automáticamente las variables que llegan a través de un formulario. Si en local estaba configurado para hacer esto y en remoto no, deberían localizarse los lugares donde se recogen las variables y utilizar las variables de entorno correctas.

Aunque este no es un caso habitual, el programador puede ponerse en contacto con el proveedor de alojamiento para ver si puede brindársele apoyo configurando el sistema o indicando los pasos a seguir para solventar en los scripts el caso.

9.1.2 Subir una base de datos al sevidor de Internet

Aparte de los archivos de la página, debe subirse la base de datos con la que se tiene que trabajar. Las bases de datos con las que trabaja PHP son muy variadas y en distintos casos puede utilizarse una u otra, por lo que los modos de subir la base de datos también pueden variar.

Es muy corriente que el proveedor de alojamiento ofrezca junto con PHP la base de datos MySQL, así que las notas para subir esa base de datos al servidor de este artículo van encaminadas a ofrecer soluciones para esa base de datos.

La base de datos MySQL no se puede subir vía FTP, como se hacía con los archivos del código PHP. Para subirla tendrán que utilizarse otros mecanismos. En esta oportunidad se van a distinguir tres casos distintos en los que el programador podría encontrarse en el momento de subir la base de datos:

A. **La base de datos que pretende subirse está vacía**. Tan sólo se han creado las tablas, pero no se han introducido datos (registros) en ellas o, a lo sumo, tienen algún dato que se ha introducido a manera de pruebas.

B. **La base de datos que se desea subir está completa y es una base de datos MySQL**. En este caso se tiene creada la base de datos en local y con toda la información dentro y, por supuesto, se requiere que esa información quede también en la base de datos remota.

C. La base de datos está **completa** (como el caso anterior), pero **no es una base de datos MySQL**. En este caso se estaría haciendo una migración de la base de datos de un sistema gestor a otro.

Se analizarán los tres casos por separado en adelante, aunque, antes de ello, se van a citar unas herramientas de considerable utilidad para la administración de cualquier base de datos remota:

9.1.2.1 PhpMyAdmin.

Una aplicación creada en PHP que podemos instalar en nuestro espacio de alojamiento para administrar la base de datos.

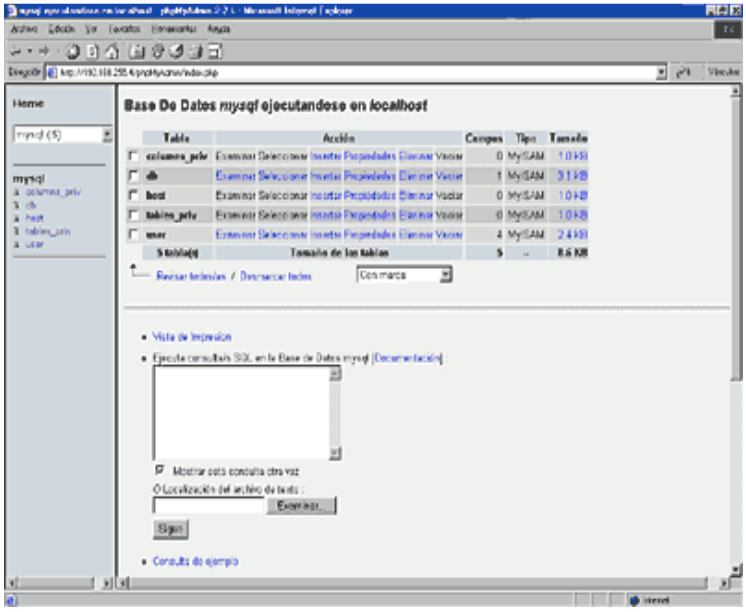
Se trata de un proyecto de código abierto en PHP para administrar bases de datos MySQL a través de una interfaz web, phpMyAdmin es un programa de libre distribución en PHP, creado por una comunidad sin ánimo de lucro. Es una herramienta muy completa que permite el acceso a todas las funciones típicas de la base de datos MySQL a través de una interfaz web muy intuitiva.

9.1.2.2 Mysql Control Center

En adelante MyCC, es una aplicación Windows que permite contectarse a múltiples bases de datos MySQL, que se encuentren en local o en remoto.

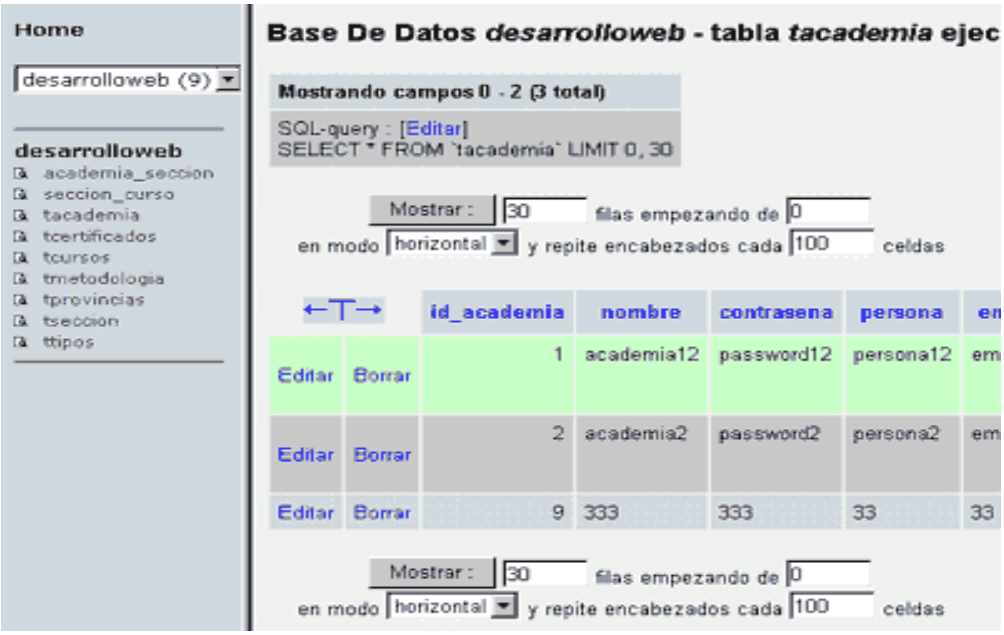
La aplicación en si no es más que un conjunto de archivos escritos en PHP que pueden copiarse en un directorio del servidor web, de modo que, cuando se tiene acceso a esos archivos, se muestran al usuario unas páginas donde se pueden encontrar las bases de datos a las que se tiene acceso el servidor de bases de datos y todas sus tablas.

Fig. 4.-Ventana de Inicio de phpMyadmin



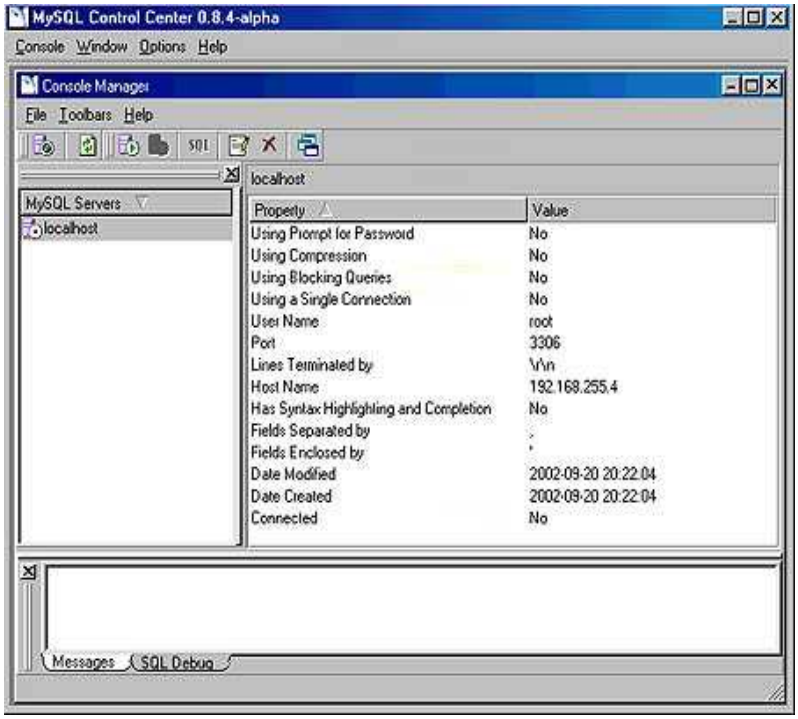
La herramienta permite crear tablas, insertar datos en las tablas existentes, navegar por los registros de las tablas, editarlos y borrarlos, borrar tablas y un largo etcétera, incluso ejecutar sentencias SQL y hacer un backup (respaldo) de la base de datos. En el apéndice de Configuración e Instalación se instruye sobre los sitios de descarga y procedimiento de instalación de ésta y otras aplicaciones de accesibilidad a mysQl y php

Fig. 5.-Entorno de Operaciones con Tablas de Bases de Datos en PhpMyadmin



Se habla en esta oportunidad de una consola de Administración de Mysql creada por Mysql AB desde la que se pueden administrar las bases de datos, los usuarios y el servidor de bases de datos, MyCC es una potente consola de Administracíon para Mysql, que permite realizar todas las tareas de administración y trabajo de Mysql mediante una interfaz gráfica, y que viene a cubrir una de las mayores objeciones que los usuarios tenían de Mysql.

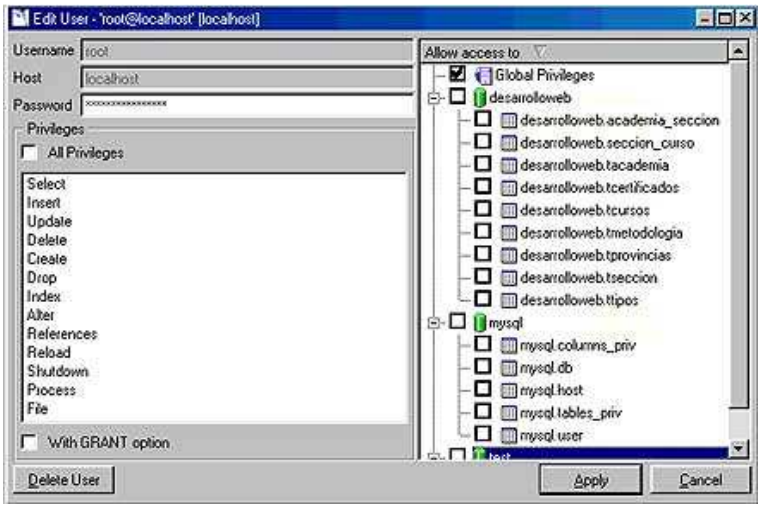
Fig. 6.-Ventana Principal de Mysql Control Center.



9.1.2.2.1 Administrador de Usuarios

Desde el se pueden añadir, borrar y modifcar las propiedades de los usarios de Mysql. Al hacer click sobre un usuario aparece la ventana de configuración del usuario.

Fig. 7.-Ventana de Administrador de Usuarios
De Mysql Control Center

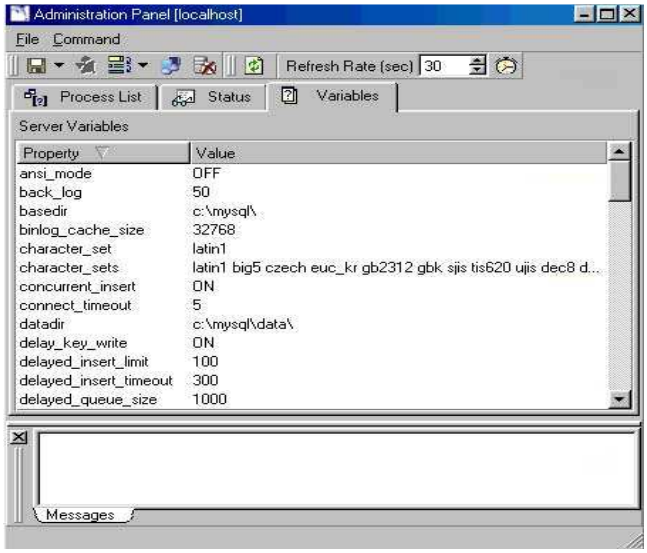


En ella se pueden configurar el nombre de usuario, la contraseña. Respecto a los permisos, permite establecer un sistema de permisos general o individualizado para cada una de las tablas. Además permite darle permiso al usuario para que se lo ceda a otros (GRANT OPTIONS). Uno de los puntos fuertes respecto a la seguridad es que al definir un usuario, hay que especificar el servidor desde al cual accederá, limitando de esta forma, los riesgos de acceso indebido.

9.1.2.2.2 Administración del servidor

La ventana de Administración del servidor consta de tres pestañas desde la cual se puede tener acceso a la lista de procesos, las variables de estado del servidor y las variables de configuración.

Fig. 8.- Ventana de Administrador de Servidor
en Mysql Control Center



Desde esta ventana además se puede hacer un Ping al servidor, detenerlo, o guardar el contenido de las variables de estado. Desde la pestaña de procesos se puede ver el estado de cada uno de los procesos activos, viendo sus propiedades y con la opción de detenerlos mediante un Kill. Desde la pestaña de estado, se puede ver el contenido de cada una de las variables de estado, y se puede personalizar para mostrar simplemente las que te interesn.

Desde la pestaña de variables se puede acceder al valor de cada una de las variables de configuración del servidor de Mysql.

9.1.2.2.3 Ventana de Base de Datos

Al seleccionar una base de datos dentro de la sección de bases de datos, aparece la ventana de base de datos, en ella se puede ver información relacionada con la base de datos, como puede ser número de tablas, el número de consultas por segundo, el tiempo que lleva en funcionamiento etc. Al seleccionar una base de datos, se pueden ver los nombres de las tablas que contiene, el número de registros que contiene.

Las tablas se pueden eliminar, vaciar y renombrar. Cuando se selecciona una tabla, se pueden ver el nombre de los campos, los tipos de datos y los parámetros opcionales de cada uno de ellos. Esta ventana permite crear nuevos campos, eliminarlos o modificar sus propiedades. Además mediante el icono de herramientas, se puede analizar la tabla, optimizarla y repararla. Al hacer doble clic sobre una tabla se muestran en la ventana los datos que contiene. Haciendo clic sobre cualquiera de los campos, se accede a la estructura de la misma. Para cada campo se puede elegir el tipo de datos, el tamaño, si es clave o no, el valor predeterminado del mismo.

Fig. 9.-Ventana de Base de Datos en Mysql Control Center

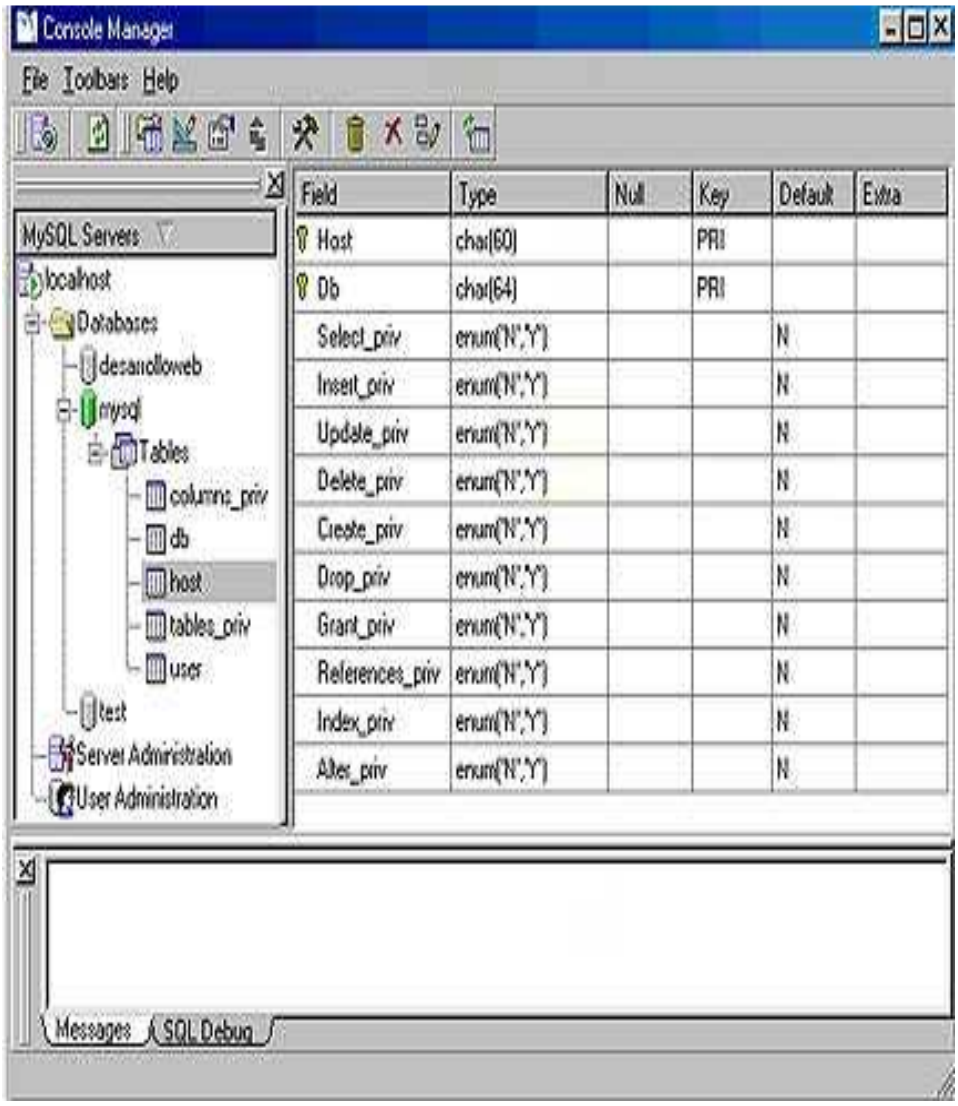
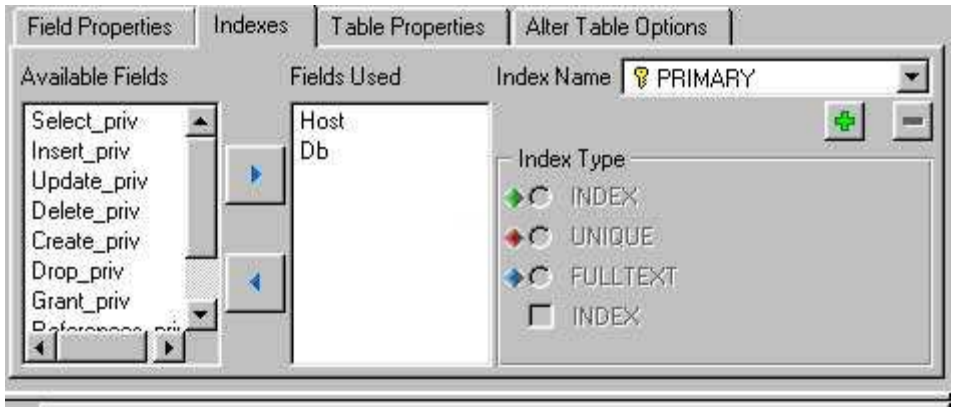


Fig. 10.-Configuración de Campos en Ventana de Base de Datos en Mysql Control Center

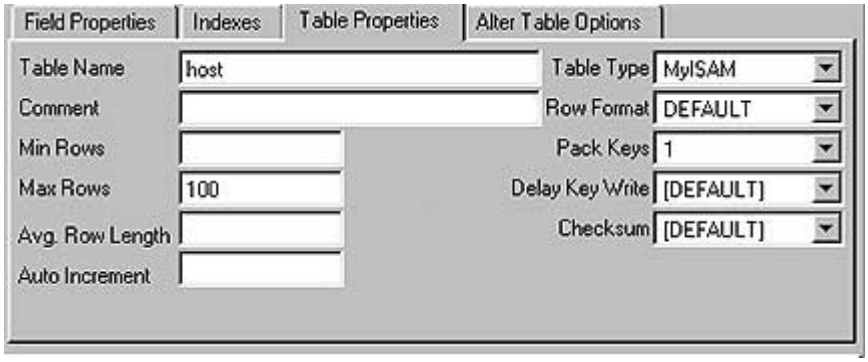
Dentro de la tabla, existe un cuadro para configurar los índices que tiene.

Fig. 11.-Configuración de Índices en Ventana de Base de Datos en Mysql Control Center



Desde ella se pueden seleccionar los campos que lo contienen, así como el tipo de índice que se desea crear. Desde la pestaña de propiedades de la tabla se acceden a los atributos de la misma, como pueden ser el tipo de tabla, el modo de escritura, el tamaño máximo de cada fila etc.

Fig. 12.-Configuración de Tablas en Ventana de Base de Datos en Mysql Control Center



9.1.2.2.4 Access.
También permite administrar una base de datos MySQL conectada en local o en remoto. En este caso se utiliza una interfaz que muchos ya conocen, como es Access, para administrar una base de datos que nada tiene que ver con dicho programa.

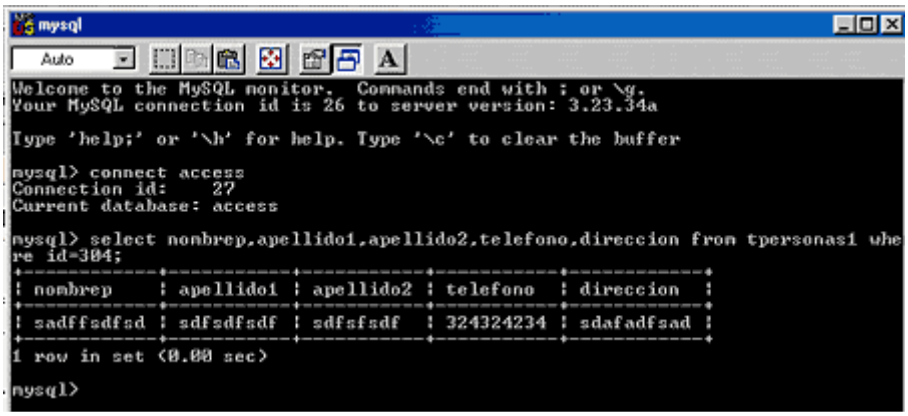
Uno de los mayores problemas que enfrenta Mysql es el de no poseer un entorno gráfico que satisfaga a la mayor parte de los usuarios. Existen magníficos proyectos a través de página

Web, como PHPmysqlAdmin, pero muchas veces existen usuarios a quienes les gustaría tener algo parecido a un Access, posibilidad de incluir formularios para la entrada de datos, o de informes sobre los datos almacenados dentro de una tabla.

Dado que no existe hoy en día una herramienta parecida al Access para trabajar con Mysql, el presente apartado expondrá la forma de trabajar con las bases de datos de Mysql utilizando el entorno gráfico de Access. Al terminarlo, el programador podrá utilizar los formularios, consultas e informes de Access con los datos de los arcivos de Mysql.

De esta forma, el encargado de actualizar los datos de una página, podrá trabajar desde Access, con la comodidad de los formularios, los menús desplegables etc., y los datos serán enviados automáticamente a Mysql.

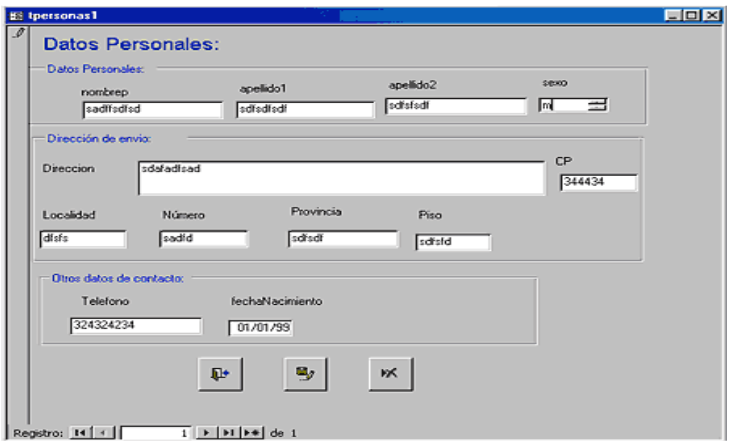
Fig. 13.-Entorno en Señal de DOS (PROMPT) Consola de mysql



Además, para la generación de formularios e informes se pueden utilizar los asistenteslo que se traduce en una tasa considerable de ahorro de tiempo.

Previamente es nacesaria la instalación de un driver para acceso a datos. Instalar este driver resulta útil para que desde un sistema Microsoft Windows se pueda obtener acceso a una base de datos MySQL. Las aplicaciones son variadas, por ejemplo puede utilizárselo para crear un DSN asociado a una base de datos MySQL, de modo que las páginas ASP podrían establecer una conexión a dicha base de datos. Otra aplicación es obtener accedso desde Access a la base de datos MySQL y exportar o importar datos (migrar los datos desde Access a MySQL y desde MySQL a Access), incluso para crear un back-end de la base de datos MySQL en interfaz Access.

Fig. 14.- Formulario de Access que ofrece acceso a una base de datos MySQL



.Primero debe descargarse la última versión de Myodbc de la página de Mysql: http://www.mysql.com/products/connector/odbc/

> **Nota:** Puede que el programador deba actualizar su sistema tenga que actualizarse. En el ordenador que tiene el sistema Windows XX y Access 2000 habría que actualizar a la versión 6 de Microsoft Jet:http://support.microsoft.com/default.aspx?scid=KB;EN-US;Q239114&

Cuando ya se tiene todo lo requerido, se procederá a instalar la actualización de Microsoft Jet, descomprimir e instalar el driver OBDC de Mysql. Cuando pregunta en la pantalla de "Data Sources" debe hacerse clic en "Close" para terminar.

Una vez se ha instalado el driver ODBC, accede al panel de control de OBDC de 32 Bits (Botón Inicio-> Configuración-> Panel de control-> Fuentes de datos ODBC 32 bits).

En este punto, el programador tendrá que elegir si quiere utilizar el driver para un solo usuario (DSN de usuario), o para cualquier usuario del ordenador (DSN de Sistema). Una vez haya elegido uno, debe hacer clic en el botón de "Agregar" para añadir una nueva fuente de datos y a continuación, selecciona el driver de Mysql. Aparecerá la siguiente pantalla:

En ella tendrán que rellenarse los siguientes campos:

Windows DSN name: Nombre de la fuente de datos que estará disponible desde Windows.

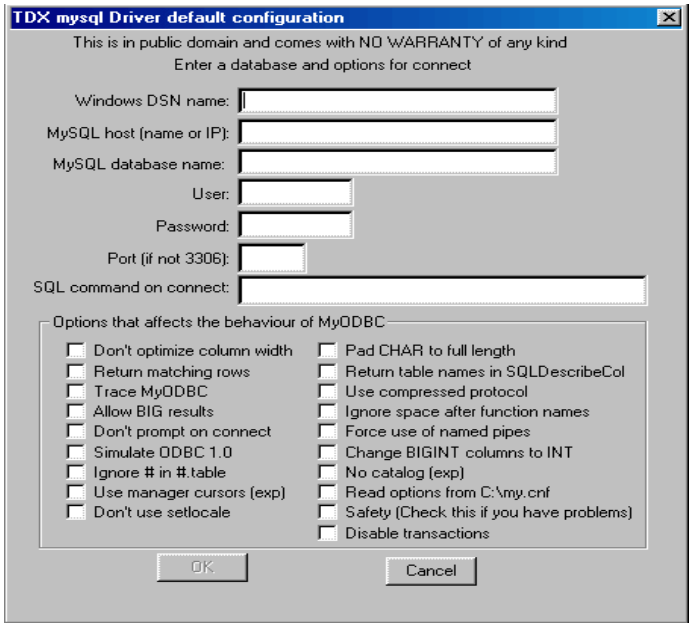
Mysql host (name or IP): Nombre o dirección IP del ordenador donde se encuentra instalado el servidor Mysql.

Mysql Database Name: Nombre de la base de datos con la que se trabajará desde la fuente de datos

User: Nombre de usuario con el que se accederá al servidor de bases de datos.

Password: Contraseña del usuario.

Fig. 15.-Ventana de Configuración de DNS en Driver para MySql

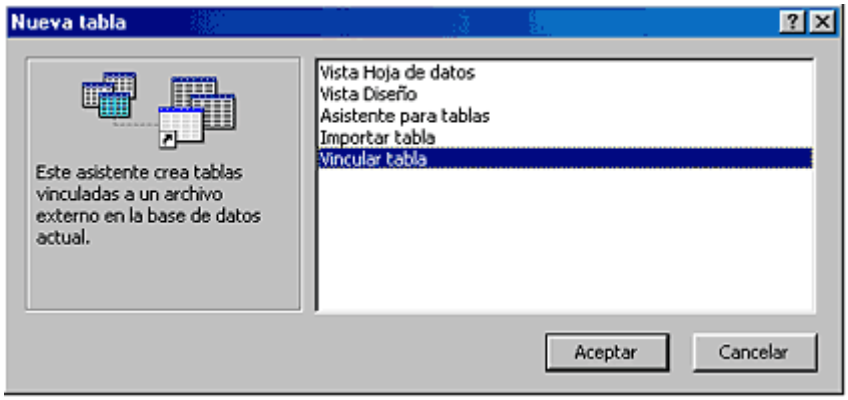


Port: Sirve para especificar el puerto en el que se encuentra el servidor Mysql, hay que poner un valor en caso de que no se esté utilizando el predeterminado, que es el 3306.

Una vez están estas opciones configuradas, se puede hacer clic en "OK" para cerrar las ventanas.

Una vez instalado el driver MyODBC instalado, lo primero que hay que hacer es crear una base de datos en blanco desde la cual se vincularán las tablas. Una vez creada, se hace clic sobre la opción de crear nueva tabla. Aparecerá la siguiente ventana en la que se seleccionará crear nueva tabla vinculada:

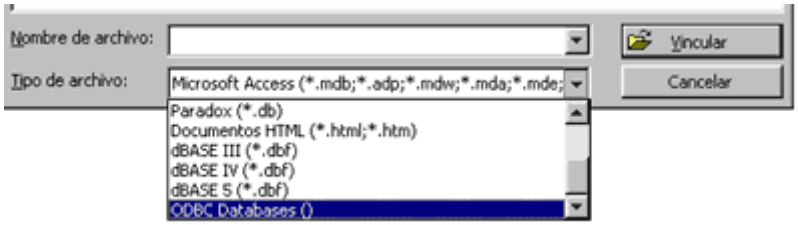
Fig. 16.- Ventana de Creación de Nueva Tabla Vinculada



Desde la ventana de vincular una tabla de una base de datos, en la parte inferior se selecciona en tipo de archivo:

9.1.2.2.5 Fuente de datos ODBC()

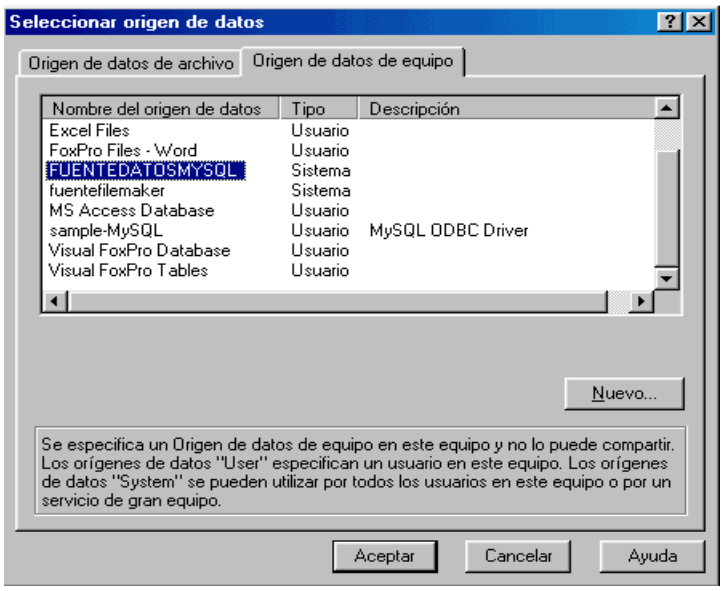
Fig. 17.- Selección de Tipo de archivo



Al hacer clic sobre Vincular, aparece la ventana para seleccionar un Origen de datos, se selecciona dentro de la pestaña de fuentes de datos del Equipo, la fuente de datos que creamos en la primera parte del artículo:

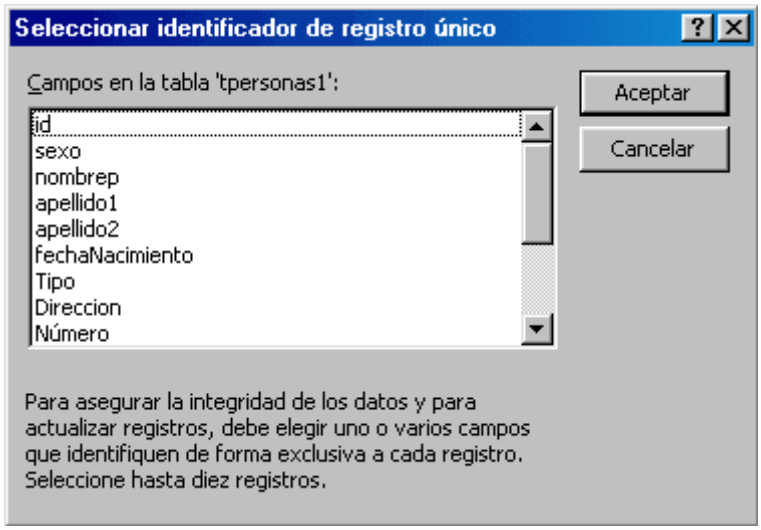
Una vez que e selecciona, se hace clic sobre Aceptar, y aparece la ventana de configuración de la fuente de datos ODBC de Mysql. Como ya esta configurada, hacemos clic sobre OK, y aparecerá la ventana en la que se pueden elegir entre las tablas que contiene la base de datos para la cual hemos configurado la fuente de datos ODBC.

Fig. 18.- Ventana de Selección de Origen de Datos



Se selecciona una tabla, y a continuación aparecerá una ventana donde deberemos especificar, hasta un máximo de diez, los campos que forman parte de la clave.

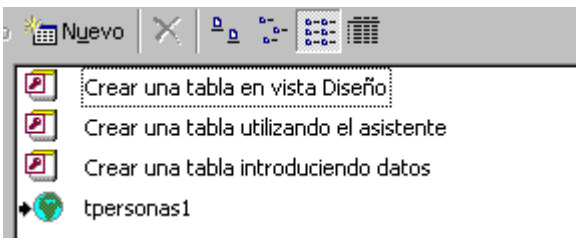
Fig. 19.- Selección de Campos de Tabla Fuente de Base de Datos



Se seleccionan los campos, y se hace clic sobre aceptar.

Aparecerá una nueva tabla dentro de la base de datos de Access. A diferencia del resto de tablas, esta no existe físicamente dentro del fichero de Access, sino que todas las modificaciones que se realizan sobre la misma, se envían a través de ODBC al fichero MYSQL.

Fig. 20.- opciones de Creación de Elementos de Access



A partir de ahora, se podrán crear formularios, consultas e informes sobre esta tabla tal y como se hace normalmente desde Access.

> **Nota:** Algunos proveedores de Hosting no incluyen en su paquete básico el acceso remoto al servidor de base de datos, o requiere de un aviso explicito por parte del cliente para su configuración.

En cualquiera de los previos tres casos lo que se le brinda al programador es la facilidad de permite ejecutar el una interfaz de administración, se trata de tareas sobre la base de datos de todo tipo, como pueden ser crear tablas, modificarlas, insertar datos, borrarlos, editarlos. Modificar o borrar tablas o campos de las mismas, etc.

La elección de una herramieta o de otra pasa por los recursos que permitan utilizaser en o proveedor escogido. Básicamente, lo que puede inducir a una opción u otra, es si permiten o no conectar de manera remota la base de datos MySQL. Existen alojamientos donde se permite esa conexión remota y otros en los cuales no.

Si no permiten conexión remota se recomienda optar por PhpMyAdmin, pues es una aplicación PHP que se conecta en local y a la que se tiene acceso desde una página web y eso lo permiten todos los proveedores, incluso hay muchos que tienen instalado ya este software para administrar las bases de datos.

En caso de que sí permitan conexión remotamente con la base de datos, se sugiere eligirMyCC o Access, que son aplicaciones Windows mucho más potentes y rápidas que las que utilizan interfaz web, como PhpMyAdmin. Es preferible utilizar MyCC porque está especialmente desarrollado para conectar y operar con bases de datos MySQL.

9.2 Subir base de datos MySQL vacía al servidor

Es muy normal que en muchas oportunidades el programador haya diseñado una base de datos para su proyecto desde cero, definiendo las distintas entidades de su modelo de datos, junto con sus campos y sus tipos.

En estos casos lo más probable es que la base de datos esté vacía, o bien contenga datos que hayan sido introducidos a modo de prueba y que no se deseen conservar cuando se suba la aplicación a Internet.

La opción más interesante entonces podría ser crear otra vez las tablas que se tienen en local en la base de datos remota. Para ello existen dos posibilidades.

a) Se tienen pocas tablas y bastante sencillas

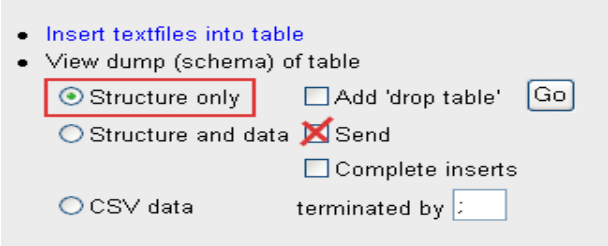
El procedimiento puede ejecutarse en remoto con alguna herramienta como PhpMyAdmin o MyCC.

b) Se tienen muchas tablas y/o tablas muy complicadas o extensas

La recomendación sería hacer un backup (respaldo) de la estructura en local y restaurarla en remoto. Esto le evitará al programador tener que volver a crear todas las tablas y definir todos sus campos y sus tipos. Puede ser un poco más complicado pero sin duda le ahorrará tiempo.

Para hacer el backup de la estructura en local puede utilizarse alguna herramienta como PhpMyAdmin, o bien utilizar el comando mysqldump desde línea de comandos de MS-DOS.

Fig. 21.- Interfaz del Procedimiento de Copia de Seguridad de Base de Datos desde PhpMyadmin



Este comando permite hacer la copia de seguridad de una o múltiples bases de datos. Además permite que estas copias de seguridad se puedan restaurar en distintos tipos de gestores de bases de datos, sin la necesidad de que se trate de un gestor de mysql. Esto lo consigue creando unos archivos, que contienen todas las sentencias sql necesarias para que sea restaurar la tabla, que incluyen desde la sentencia de creación de la tabla, hasta una sentencia insert por cada uno de los registros que forman parte de la misma.

El comando dispone de una amplia variedad de opciones que le permitirán realizar la copia de la forma más conveniente para el propósito de la misma. Para poder restaurar la copia de seguridad, bastará con ejecutar todas las sentencias sql que se encuentran dentro del fichero, bien desde la línea de comandos de mysql, o desde la pantalla de creación de sentencias sql de cualquier entorno gráfico como puede ser el Mysql Control Center.

Las limitaciones de la restauración dependerán de las opciones que se han especificado a la hora de hacer la copia de seguridad, por ejemplo, si se incluye la opción --add-drop-table al hacer la copia de seguridad, se podrán restauran tablas que existen actualmente en el servidor (borrándolas primero). Por lo que es necesario estudiar primero los procedimientos que se utilizarán tanto en la copia como en la restauración, para que todo salga correcto.

Algunas de las opciones que tiene son:

--add-locks : Añade LOCK TABLES antes, y UNLOCK TABLE despues de la copia de cada tabla.

--add-drop-table: Añade un drop table antes de cada sentencia create

-A, --all-databases: Copia todas las bases de datos. Es lo mismo que utilizar --databases seleccionando todas.

-a, --all: Incluye todas las opciones de creación específicas de Mysql.

--allow-keywords: Permite la creación de nombes de columnas que son palabras clave, esto se realiza poniendo de prefijo a cada nombre de columna, el nombre de la tabla

-c, --complete-insert: Utiliza inserts incluyendo los nombres de columna en cada sentencia (incrementa bastante el tamaño del fichero)

-C, --compress: Comprime la información entre el cliente y el servidor, si ambos soportan compresión.

-B, --databases: Para copiar varias bases de datos. En este caso, no se especifican tablas. El nombre de los argumentos se refiere a los nombres de las bases de datos. Se incluirá USE db_name en la salida antes de cada base de datos.

--delayed: Inserta las filas con el comando INSERT DELAYED.

-e, --extended-insert: Utiliza la sintaxis de INSERT multilinea. (Proporciona sentencias de insert más compactas y rápidas.)

-#, --debug[=option_string]: Utilización de la traza del programa (para depuración).

--help: Muestra mensaje de ayuda y termina.

--fields-terminated-by=...
--fields-enclosed-by=...
--fields-optionally-enclosed-by=...
--fields-escaped-by=...
--lines-terminated-by=...

Las anteriores variantes se utilizan con la opción -T y tienen el mismo significado que la correspondiente cláusula LOAD DATA INFILE.

-F, --flush-logs: Escribe en disco todos los logs antes de comenzar con la copia

-f, --force,: Continúa aunque se produzca un error de SQL durante la copia.

-h, --host=..: Copia los datos del servidor de Mysql especificado. El servidor por defecto es localhost.

-l, --lock-tables.: Bloquea todas las tablas antes de comenzar con la copia. Las tablas se bloquean con READ LOCAL para permitir inserts concurrentes en caso de las tablas MyISAM. Cuando se realiza la copia de múltiples bases de datos, --lock-tables bloqueará la copia de cada base de datos por separado. De forma que esta opción no garantiza que las tables serán consistentes lógicamente entre distintas bases de datos. Las tablas en diferentes bases de datos se copiarán en estados completamente distintos.

-K, --disable-keys: Se incluirá en la salida /*!40000 ALTER TABLE tb_name DISABLE KEYS */; y /*!40000 ALTER TABLE tb_name ENABLE KEYS */; Esto hará que carga de datos en un servidor MySQL 4.0 se realice más rápido debido a que los índices se crearán después de que todos los datos hayan sido restaurados.

-n, --no-create-db: No se incluirá en la salida CREATE DATABASE /*!32312 IF NOT EXISTS*/ db_name; Esta línea se incluye si la opción --databases o --all-databases fue seleccionada.

-t, --no-create-info: No incluirá la información de creación de la tabla (sentencia CREATE TABLE).

-d, --no-data: No incluirá ninguna información sobre los registros de la tabla. Esta opción sirve para crear una copia de sólo la estructura de la base de datos.

--opt: Lo mismo que --quick --add-drop-table --add-locks --extended-insert --lock-tables. Esta opción le debería permitir realizar la copia de seguridad de la base de datos de la forma más rápida y efectiva.

-pyour_pass, --password[=your_pass]: Contraseña utilizada cuando se conecta con el servidor. Si no se especifica, `=your_pass', mysqldump preguntará la contraseña.

-P, --port=...: Puerto utilizado para las conexiones TCP/IP

--protocol=(TCP | SOCKET | PIPE | MEMORY): Especifica el protocolo de conexión que se utilizará.

-q, --quick: No almacena en el buffer la sentencia, la copia directamente a la salida. Utiliza mysql_use_result() para realizarlo.

-Q, --quote-names: Entrecomilla las tablas y nombres de columna con los caracteres ``'.

-r, --result-file=...: Redirecciona la salida al fichero especificado. Esta opción se debería utilizar en MSDOS, porque previene la conversión de nueva línea `\n' en nueva línea y retorno de carro`\n\r'.

--single-transaction: Utiliza el comando BEGIN antes de realizar la copia desde el servidor. Es muy útil con las tables InnoDB y el nivel de transacción READ_COMMITTED, porque en este modo realizará la copia de seguridad en un estado consistente sin necesidad de bloquear las aplicaciones. Consultar el manual para más detalles.

-S /path/to/socket, --socket=/path/to/socket: El archivo de sockets que se especifica al conectar al localhost (que es el host predeterminado).

--tables: sobreescribe la opción --databases (-B).

-T, --tab=path-to-some-directory: Crea un archivo table_name.sql, que contiene la sentencia de creación de SQL, y un fichero table_name.txt, que contiene los datos de cada tabla. El formato del fichero `.txt' se realiza de acuerdo con las opciones --fields-xxx y --lines--xxx options. Nota: Esta opción sólo funciona si el comando mysqldump se ejecuta en la misma másquina que el demonio mysqld, el usuario deberá tener permisos para crear y escribir el fichero en la ubicación especificada.

-u nombre_usuario, --user=nombre_usuario: El nombre de usuario que se utilizará cuando se conecte con el servidor, el valor predeterminado es el del usuario actual.

-v, --verbose: Va mostrando información sobre las acciones que se van realizando (más lento)

-w, --where='cláusula where': Sirve para realizar la copia de determinados registros

-X, --xml: Realiza la copia de seguridad en un documento xml

-x, --first-slave: Bloquea todas las tablas de todas las bases de datos

9.2.1 Ejemplos de comandos mysqldump

Para realizar la copia se seguridad de la base de datos mibase al fichero copia_seguridad.sql

```
mysqldump --opt mibase > copia_seguridad.sql
```

Otro ejemplo más complejo de comando mysqldump para hacer el backup de una base de datos es el siguiente:

```
mysqldump --opt --password=miclave --user=miuser mibasededatos > archivo.sql
```

En este último caso se está indicando un nombre de usuario y una clave para obtener acceso a la base de datos sobre la que se está haciendo el backup: mibasededatos. Las sentencias SQL para reconstruir esa base de datos se volcarán en el fichero archivo.sql.

9.2.2 Restaurar la base de datos

Si se desea recuperar la información de un fichero para restaurar una copia de seguridad de la base de datos lo haremos con el comando mysql. Utilizaremos una sintaxis como esta:

```
mysql mibase < archivo.sql
```

En este ejemplo se restauraría la base de de datos mibase con el backup almacenado en el fichero archivo.sql.

Otro ejemplo más complejo de comando para restaurar una base de datos es el siguiente:

```
mysql --password=miclave --user=miuser mibase < archivo.sql
```

Es el mismo ejemplo que el anterior, pero indicando un nombre de usuario y una clave con las que acceder a la base de datos mibase.

9.3 Subir una base de datos MySQL con la estructura y los datos

Si la base de datos que desea subirse está llena de información y el programador desea que se conserve, una vez subida la base de datos a remoto, es necesario realizar un backup de la base de datos y restaurarlo en remoto.

> **Nota:** Estas recomendaciones están pensadas para subir una base de datos MySQL que se puedan tener en local a una base de datos MySQL que se haya contratado en remoto. Si la base origen no es MySQL se estaría hablando de una migración de bases de datos.

En este caso el procedimiento sería muy parecido al de subir una base de datos vacía, descrito anteriormente, con la salvedad de que ahora debe extraerse no solo la estructura de la base de datos, sino también los registros que contiene.

Para ello puede utilizarse mysqldump, o bien PhpMyAdmin, seleccionando la opción que indica que el backup contenga la estructura y los datos “Structure and data” (en versiones en inglés).

La estructura y los datos vendrán en un archivo de texto con una serie de sentencias SQL para crear las tablas y los insert necesarios para introducir cada uno de los datos.

El procedimiento de restaurar la base de datos se desarrollará tal como se ha descrito para el caso de que la base de datos estuviera vacía, con la ayuda de una instalación de PhpMyAdmin en remoto o un MyCC que se conecte a la base de datos contratada en el servidor de Internet.

Si se presentan problemas para subir el archivo de backup de la base de datos es posible que en el proveedor contratado de alojamiento se pueda encontrar ayuda a subir el fichero y restaurarlo. Como el proveedor dispone de los servidores en sus propias instalaciones, tiene muchas más posibilidades que el programador para trabajar con las bases de datos, sin temor a que las lentas comunicaciones por Internet arrojen errores en la restauración de los datos.

Si el proveedor no puede ayudar al programado, seguramente disponga y le indique a éste algún mecanismo para realizar la tarea sin lugar a errores. Puede ocurrir con algún proveedor que diga que se encarga de todo pero exija el pago de las horas de trabajo del informático que va a restaurar el backup de la base de datos. Si no pone facilidades ni siquiera en esto posiblemente sea mejor optar por otro proveedor porque su servicio no sería el más adecuado.

9.4 Migrar una base de datos a MySQL

El último caso en el que el programador podrá toparse a la hora de subir una base de datos al servidor del proveedor de alojamiento es que la base de datos la tenga creada en local, pero en un sistema gestor distinto del que va a utilizarse en remoto. En remoto se asume siempre que se va a utilizar la base de datos MySQL. En local podría disponerse de una base de datos Access, SQL Server o de otro sistema de base de datos.

El proceso de la migración puede ser bastante complejo y, como existe una amplia gama de bases de datos distintas, puede resultar algo difícil de dar una indicación que funcione en todos los casos. Además, aparte de la dificultad de transferir la información entre los dos sistemas gestores de base de datos, también influirá mucho en la complejidad del problema el tipo de datos de las tablas que se están utilizando. Por ejemplo, las fechas, los campos numéricos con decimales o los booleanos pueden dar problemas al pasar de un sistema a otro porque pueden almacenarse de maneras distintas o, en el caso de los números, con una precisión distinta.

9.4.1. Recomendaciones para migrar de Access a MySQL

Si la base de datos anterior estaba construida en Access el proceso podría resultar muy fácil, gracias a que MySQL dispone de un driver ODBC para sistemas Windows, que le permite conectar Access con el propio MySQL y efectuar transferencia de data fácilmente.

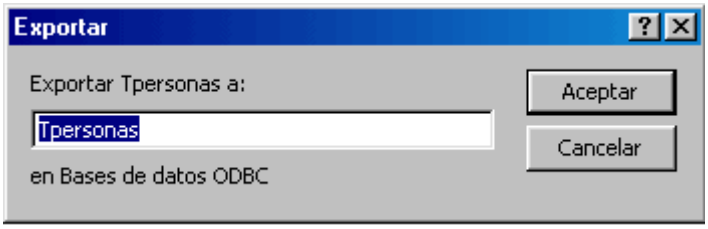
No es de extrañar que un programador web haya iniciado sus pasos en la web sirviéndonos de una base de datos sencilla como Access. Tampoco es de extrañar que, llegado el momento, se vea en la necesidad de migrar a herramientas más serias y opte a por servidor de datos como MySQL. Aquí se muestra una manera bastante práctica de migrar los datos de la una a la otra.

> **Referencia:** Para realizar esta tarea es necesario que el programador haya descargado e instalado el driver ODBC y en su sistema Windows.

Para exportar una tabla a Mysql, hay que abrir la base de datos y seleccionar la tabla. Después, seguir la ruta de menú Archivo->Exportar. En la pantalla de exportar, en la opción Guardar como tipo, seleccionar ODBC databases().

Una vez se ha hecho esto, aparece una ventana que solicita al usuario el nombre que se le desea dar a la tabla en Mysql, por defecto aparece el mismo.

Fig. 22.- Asignación de Nombre al elemento de Exportación de Base de Datos



Posteriormente debe Hacerse clic sobre "Aceptar", y aparecerá la pantalla en la que se pide que selecciones el origen de datos ODBC:

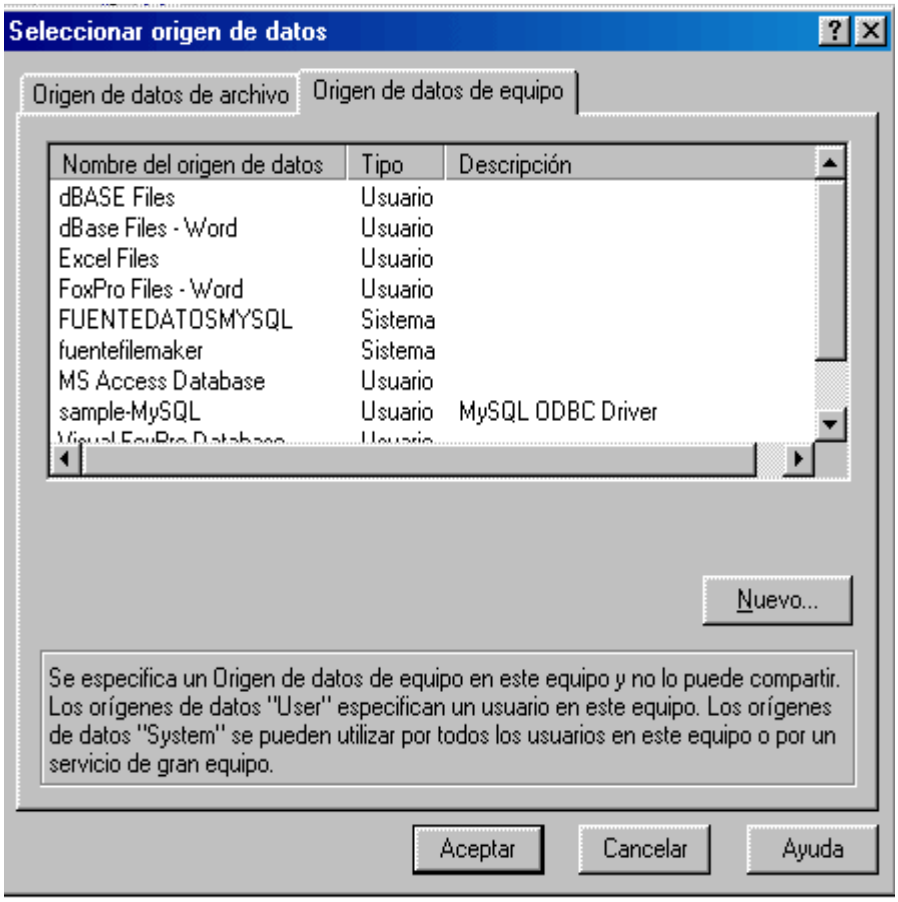
> **Nota:** pudiera ser en algún caso que los tipos de los datos de la base en los sistemas MySQL y Access no sean totalmente compatibles y se produzca alguna anomalia al exportarlos. Realmente es una posibilidad

latente, aunque en las pruebas que se han llevado a cabo no se ha observado ningún tipo de problema, bien es cierto que los campos que se han trabajado no eran muy raros.

Seleccionar origen de datos de equipo, y dentro de esta el nombre de la fuente de datos que se han creado anteriormente. Una vez se haya seleccionado y Aceptado, aparecerá la pantalla de configuración del driver por si se desea marcar para esta acción en concreto algunas de las opciones de configuración que aparecen en el driver ODBC. Si no se desea marcar ninguna, sencillamente se hace clic sobre "OK" y los datos comenzarán a exportarse.

La idea en este último caso es instalar MySQL en local y realizar la migración desde Access en local a MySQL en local y luego podría hacerse un backup de la base de datos local y subirla a remoto, tal y como se ha descrito en apartados anteriores.

Fig. 22.- Asignación de Nombre al elemento de Exportación de Base de Datos



9.4.2 Otras bases de datos u otras técnicas

Si la base de datos origen dispone de un driver ODBC no habrá (en teoría) problema para conectarla con Access, de manera similar a como se conecta con MySQL. Entonces podrías utilizarse Access como puente para exportar los datos, porque desde allí se podría tener acceso a los dos sistemas gestores de bases de datos.

Si no se dispone de Access, o la base de datos original no tiene driver ODBC, o bien no se desarrolla correctamente el proceso, otra posibilidad es exportar los datos a archivos de texto, separados por comas. Muchas bases de datos tienen herramientas para exportar los datos de

las tablas a archivos de texto, los cuales se pueden luego ser introducidos por el programador en su sistema gestor destino (MySQL) con la ayuda de alguna herramienta como PhpMyAdmin.

Para ello, en la página de propiedades de la tabla se encontrará una opción para hacer el backup de la tabla y para introducir archivos de texto dentro de una tabla "Insert textfiles into table" en (versiones en inglés).

Teniendo Acceso a ese enlace podrá verse un formulario donde introducir las características del archivo de texto, como por ejemplo el carácter (signo) utilizado como separador de campos, o el terminador de líneas, etc, junto con el propio archivo con los datos, y PhpMyAdmin se encargará de todo el trabajo de incluir esos datos en la tabla.

Como se habrá supuesto, es necesario tener creada la tabla en remoto para que puedan introducírsele los datos del archivo de texto.

Fig. 23.- Generación de Archivo de Texto para Migración De Base de Datos en PhpMyadmin.

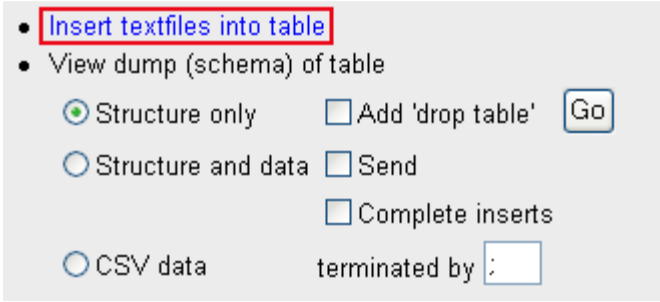
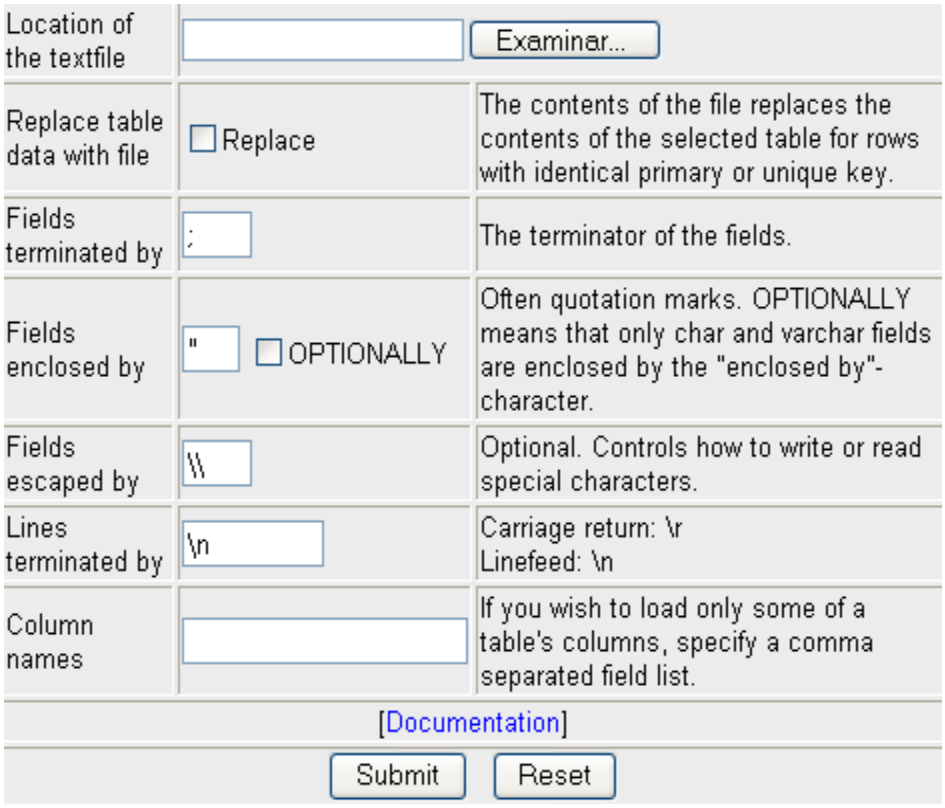


Fig. 24.-Procedimiento de Configuración de Generación de Archivo de Texto para Migración de Base de Datos en PhpMyadmin.



9.4.3 Cambios de un formato de datos a otro

Toda la migración debe tener en cuenta muy especialmente, como ya se señaló, las maneras que tenga cada base de datos de guardar la información, es decir, del formato de sus tipos de datos. Es necesario contar siempre con la posible necesidad de transformar algunos datos como pueden ser los campos booleanos, fechas, campos memo (texto con longitud indeterminada), etc, que pueden almacenarse de maneras distintas en cada uno de los sistemas gestores, origen y destino.

En algunos casos posiblemente sea oportuno que realizar algún script que realice los cambios necesarios en los datos. Por ejemplo puede ser para localizar los valores booleanos, guardados como true / false a valores enteros 0 / 1, que es como se almacena en MySQL. También las fechas pueden sufrir cambios de formato, mientras que en Access aparecen en castellano (dd/mm/aaaa) en MySQL aparecen en el formato aaaa-mm-dd. PHP puede ayudar al programador web en la tarea de hacer este script, también Visual Basic Script para Access puede hacer estas tareas complejas y el propio lenguaje SQL, a base de sentencias dirigidas contra la base de datos, puede servir para algunas acciones sencillas.

10. Programación orientada a objetos en PHP

La programación orientada a objetos es una metodología de programación avanzada y bastante extendida, en la que los sistemas se modelan creando clases, que son un conjunto de datos y funcionalidades. Las clases son definiciones, a partir de las que se crean objetos. Los objetos son ejemplares de una clase determinada y como tal, disponen de los datos y funcionalidades definidos en la clase.

La programación orientada a objetos permite concebir los programas de una manera bastante intuitiva y cercana a la realidad. La tendencia es que un mayor número de lenguajes de programación adopten la programación orientada a objetos como paradigma para modelar los sistemas. Prueba de ello es la nueva versión de PHP (5), que implanta la programación de objetos como metodología de desarrollo. También Microsoft ha dado un vuelco hacia la programación orientada a objetos, ya que .NET dispone de varios lenguajes para programar y todos orientados a objetos.

Así pues, la programación orientada a objetos es un tema de gran interés, pues es muy utilizada y cada vez resulta más esencial para poder desarrollar en casi cualquier lenguaje moderno. En este apartado se analizarán algunas nociones sobre la programación orientada a objetos en PHP. Aunque es un tema bastante amplio, novedoso para muchos y en un principio, difícil de asimilar, se intentará explicar la sintaxis básica de PHP para utilizar objetos, sin profundizar en mucha teoría de programación orientada a objetos en general.

10.1 Las clases: class

Una clase es un conjunto de variables, llamadas atributos, y funciones, analizadas como métodos, que trabajan sobre esas variables. Las clases son, al fin y al cabo, una definición: una especificación de propiedades y funcionalidades de elementos que van a participar en programas.

Por ejemplo, la clase "Caja" tendría como atributos características como las dimensiones, color, contenido y propiedades semejantes. Las funciones o métodos que podrían incorporarse a la clase "caja" son las funcionalidades que se desea que realice la caja, como introduce(), muestra_contenido(), comprueba_si_cabe(), vaciate(), etc.

Las clases en PHP se definen de la siguiente manera:

```
<?
class Caja{
   var $alto;
   var $ancho;
```

```
    var $largo;
    var $contenido;
    var $color;

function introduce($cosa){
    $this->contenido = $cosa;
}

function muestra_contenido(){
    echo $this->contenido;
}
}
?>
```

En este ejemplo se ha creado la clase Caja, indicando como atributos el ancho, alto y largo de la caja, así como el color y el contenido. Se han creado, para empezar, un par de métodos, uno para introducir un elemento en la caja y otro para mostrar el contenido.

Si nos fijamos, los atributos se definen declarando unas variables al principio de la clase. Los métodos se definen declarando funciones dentro de la clase. La variable $this, utilizada dentro de los métodos la explicaremos un poco más abajo.

10.1.1 Utilizar la clase

Las clases solamente son definiciones. Si se desea utilizar la clase es necesario crear un ejemplar de dicha clase, lo que corrientemente se le llama instanciar un objeto de una clase.

```
$micaja = new Caja;
```

Con esto se ha creado creado, o mejor dicho, instanciado, un objeto de la clase Caja llamado $micaja.

```
$micaja->introduce("algo");
$micaja->muestra_contenido();
```

Con estas dos sentencias se esta introduciendo "algo" en la caja y luego mostrando ese contendido en el texto de la página. Nótese que los métodos de un objeto se llaman utilizando el código "->".

```
nombre_del_objeto->nombre_de_metodo()
```

El obtener acceso a los atributos de una clase también se hace con el código "->". De esta forma:

```
nombre_del_objeto->nombre_del_atributo
```

10.1.2 La variable $this

Dentro de un método, la variable $this hace referencia al objeto sobre el que se invoca el método. En la invocación $micaja->introduce("algo") se está llamando al método introduce sobre el objeto $micaja. Cuando se está ejecutando ese método, se vuelca el valor que recibe por parámetro en el atributo contenido. En ese caso $this->contenido hace referencia al atributo contenido del objeto $micaja, que es sobre el que se invocaba el método.

10.2 Constructores en PHP

Los constructores son funciones, o métodos, que se encargan de realizar las tareas de inicialización de los objetos al ser instanciados. Es decir, cuando se crean los objetos a partir de las clases, se llama a un constructor que se encarga de inicializar los atributos del objeto y realizar cualquier otra tarea de inicialización que sea necesaria.

No es obligatorio disponer de un constructor, pero resultan muy útiles y su uso es muy habitual. En el ejemplo de la caja, que se comentaba en el apartado anterior, lo normal sería inicializar las variables como color o las relacionadas con las dimensiones y, además, indicar

que el contenido de la caja está vacío. Si no hay un constructor no se inicializan ninguno de los atributos de los objetos.

El constructor se define dentro de la propia clase, como si fuera otro método. El único detalle es que el constructor debe tener el mismo nombre que la clase. Es necesario atender al hecho de PHP, que diferencia entre mayúsculas y minúsculas.

Para la clase Caja definida anteriormente, se podría declarar este constructor:

```
function Caja($alto=1,$ancho=1,$largo=1,$color="negro"){
   $this->alto=$alto;
   $this->ancho=$ancho;
   $this->largo=$largo;
   $this->color=$color;
   $this->contenido="";
}
```

En este constructor se reciben por parámetro todos los atributos que hay que definir en una caja. Es muy útil definir unos valores por defecto en los parámetros que recibe el constructor, igualando el parámetro a un valor dentro de la declaración de parámetros de la función constructora, pues así, aunque se llame al constructor sin proporcionar parámetros, se inicializará con los valores por defecto que se hayan definido.

Es importante señalar que en los constructores no se tienen por qué recibir todos los valores para inicializar el objeto. Hay algunos valores que pueden inicializarse a vacío o a cualquier otro valor fijo, como en este caso el contenido de la caja, que inicialmente hemos supuesto que estará vacía.

10.3 Herencia en PHP

La programación orientada a objetos tiene un mecanismo llamado herencia por el que se pueden definir clases a partir de otras clases. Las clases realizadas a partir de otra clase o mejor dicho, que extienden a otra clase, se llaman clases extendidas o clases derivadas.

Las clases extendidas heredan todos los atributos y métodos de la clase base. Además, pueden tener tantos atributos y métodos nuevos como se desee.

Para ampliar el ejemplo que venimos desarrollando, la clase Caja, vamos a crear una clase extendida llamada Caja_tematica. Esta clase hereda de caja, pero además tiene un "tema", que es la descripción del tipo de cosas que metemos en la caja. Con esto podemos tener varias cajas, cada una con cosas de un tema concreto.

```
class Caja_tematica extends Caja{
   var $tema;

   function define_tema($nuevo_tema){
   $this->tema = $nuevo_tema;
   }
}
```

En esta clase heredamos de Caja, con lo que se tiene a disposición todos los atributos y métodos de la clase base. Además, se ha definido un nuevo atributo, llamado $tema, y un método, llamado define_tema(), que recibe el tema con el que se desea etiquetar la caja.

Podría utilizarse la clase Caja_tematica de manera similar a como lo hacíamos con la clase Caja original.

```
$micaja_tematica = new Caja_tematica();
$micaja_tematica->define_tema("Cables y contectores");
$micaja_tematica->introduce("Cable de red");
$micaja_tematica->introduce("Conector RJ45");
```

```
$micaja_tematica->muestra_contenido();
```

En este caso, el resultado que se obtiene es parecido al que se obtiene para la clase base. Sin embargo, cuando se muestra el contenido de una caja, lo más interesante sería que se indicara también el tipo de objetos que contiene la caja temática. Para ello, tenemos que redefinir el método muestra_contenido().

10.3.1 Redefinir métodos en clases extendidas

Redefinir métodos significa volver a codificarlos, es decir, volver a escribir su código para la clase extendida. En este caso, tenemos que redefinir el método muestra_contenido() para que muestre también el tema de la caja.

Para redefinir un método, lo único que debemos hacer es volverlo a escribir dentro de la clase extendida.

```
function muestra_contenido(){
   echo "Contenido de la caja de <b>" . $this->tema . "</b>: " . $this->contenido;
}
```

En este ejemplo hemos codificado de nuevo el método entero para mostrar los datos completos.

En algunas ocasiones es muy útil apoyarse en la definición de un método de la clase base para realizar las acciones de la clase extendida. Por ejemplo, para este ejemplo, tenemos que definir un constructor para la clase Caja_tematica, en el que también se inicialice el tema de la caja. Como ya existe un método constructor en la clase base, no merece la pena reescribir el código de éste, lo mejor es llamar al constructor que había definido en la clase Caja original, con lo que se inicializarán todos los datos de la clase base, y luego realizar la inicialización para los atributos de la propia clase extendida.

Para llamar a un método de la clase padre dentro del código de un método que estamos redefiniendo, utilizamos una sintaxis como esta:

```
function Caja_tematica($alto=1,$ancho=1,$largo=1,$color="negro",$tema="Sin clasificación"){
   parent::Caja($alto,$ancho,$largo,$color);
   $this->tema=$tema;
}
```

Aquí se observa la redefinición del constructor, de la clase Caja, para la clase Caja_tematica. El constructor hace primero una llamada al constructor de la clase base, a través de una referencia a "parent". Luego inicializa el valor del atributo $tema, que es específico de la Caja_tematica.

En la misma línea de trabajo, puede redefinirse el método muestra_contenido() apoyándonos en el que fue declarado en la clase base. El código quedaría como sigue:

```
function muestra_contenido(){
   echo "Contenido de la caja de <b>" . $this->tema . "</b>: ";
   parent::muestra_contenido();
}
```