UTN.BA FACULTAD REGIONAL BUENOS AIRES
SECRETARÍA DE EXTENSIÓN UNIVERSITARIA FRBA UTN

Centro de e-Learning

EXPERTO UNIVERISTARIO EN MySQL Y PHP NIVEL INICIAL

www.sceu.frba.utn.edu.ar/e-learning

Módulo 1

INTRODUCIÉNDONOS EN MySQL Y PHP

Arquitectura Cliente-Servidor.

Introducción a PHP. Sintaxis básica.

Presentación de la Unidad:

Marcaremos la diferencia de las páginas estáticas vs. Las páginas dinámicas.

Haremos una introducción al lenguaje de programación PHP. Veremos como funciona, como se configura, como es su funcionamiento y para que nos sirve utilizar PHP.

Cual es su sintaxis y como se intercala dentro del codigo HTML.

Objetivos:

- Aprender cuales son las alternativas de configuración del lenguaje PHP.
- Aprender como es su funcionamiento.
- Aprender la sintaxis del mismo y como se intercalan las instrucciones dentro del código HTML.

Temario:

PHP

Conceptos Básicos.

Concepto de páginas dinámicas.

Comencemos refiriéndonos al leguaje HTML. HTML no es lenguaje de programación sino, más bien, se trata de un lenguaje descriptivo que tiene como objeto dar formato al texto y las imágenes que pretendemos visualizar en el navegador.

A partir de este lenguaje somos capaces de introducir enlaces, seleccionar el tamaño de las fuentes o intercalar imágenes, todo esto de una manera prefijada y en ningún caso inteligente.

En efecto, el HTML no permite realizar un simple cálculo matemático o crear una página de la nada a partir de una base de datos. A decir verdad, el HTML, aunque muy útil a pequeña escala, resulta bastante limitado a la hora de concebir grandes sitios o portales.

Es esta deficiencia del HTML la que ha hecho necesario el empleo de otros lenguajes accesorios mucho más versátiles y de un aprendizaje relativamente más complicado, capaces de responder de manera inteligente a las demandas del navegador y que permiten la automatización de determinadas tareas tediosas e irremediables como pueden ser las actualizaciones, el tratamiento de pedidos de una tienda virtual.

Estos lenguajes capaces de recrear a partir de ciertos "scripts" un sinfín de páginas automatizadas son los protagonistas de este concepto de páginas dinámicas.

Páginas dinámicas vs HTML

A pesar de que las páginas dinámicas nos puedan en un principio limitar a causa de su mayor complejidad con respecto al HTML, todas las ventajas que nos ofrecen compensan con creces este esfuerzo inicial.

No obstante, hay que ser consciente del posible interés que pueda tener para uno el lanzarse en esta aventura de aprender un nuevo lenguaje y volver a rediseñar su propio sitio.

Si la página en la que estamos pensando o que queremos rediseñar es relativamente pequeña, no necesita estar al día continuamente sino que sus contenidos son perennes y no hemos previsto el pagar por mantenerla, el empleo de páginas dinámicas puede quedarse grande y resultar a todas luces improductivo.

Por el contrario, si el sitio es extenso y sus contenidos cambian rápidamente, nos interesa el automatizar en la medida de lo posible todas las tareas de tal forma que podamos gestionar su explotación de la manera más óptima.

Para dejar más claro hasta qué punto resulta útil utilizar páginas dinámicas lo mejor será ejemplificarlo a partir de un sitio web modelo.

Supongamos que hemos decidido realizar un portal de televisión donde una de las informaciones principales a proveer podría ser la programación semanal. Efectivamente,

esta información suele ser dada por las televisiones con meses de antelación y podría ser muy fácilmente almacenada en una base de datos. Si trabajásemos con páginas HTML, tendríamos que construir una página independiente para cada semana en la cual introduciríamos "a mano" cada uno de los programas de cada una de las cadenas.

Asimismo, cada semana nos tendríamos que acordar de descolgar la página de la semana pasada y colgar la de la anterior.

Todo esto podría ser fácilmente resuelto mediante páginas dinámicas. En este caso, lo que haríamos sería crear un programa (solo uno) que se encargaría de recoger de la base de datos de la programación aquellos programas que son retransmitidos en las fechas que nos interesan y de confeccionar una página donde aparecerían ordenados por cadena y por hora de retransmisión. De este modo, podemos automatizar un proceso y desentendernos de un aspecto de la página por unos meses.

Este hecho lo podríamos aplicar a otras situaciones: podemos preparar el horóscopo de todos los días, las promociones de un sitio de comercio electrónico, etc.

Además, tampoco resultaría complicado el introducir una pequeña caja de búsqueda que nos permitiera dar rápidamente con el programa que queremos ver, saber a qué hora y en qué cadena se emite.

Volviendo a nuestro portal de televisión, en él hay una sección en la cual presentamos todas las
series actualmente emitidas con comentarios sobre ella, fotos, etc. Podríamos, en lugar de hacer una
página HTML por serie, hacer una única página dinámica en contacto con una base de datos en la cual visualizamos las fotos y comentarios relativos a la serie que nos interesa.

Asimismo, si lo que buscamos es modificar el formato del texto de dicha sección, podemos automatizar este proceso sin necesidad de cambiar a mano cada una de las etiquetas.

Otra serie de aspectos tales como la gestión de las lenguas, podrían ser fácilmente resueltos sin para ello duplicar el número de páginas y buscar los textos a traducir penosamente entre el código HTML.

¿Qué es PHP?

Las siglas “PHP” significan “PHP: Hypertext Processor” algunos también lo traducen como “Pre- Hypertext Processor” (Procesador Previo al Hipertexto). Es fin, lo importante es que es un lenguaje de programación que sirve para hacer páginas de Internet dinámicas, con la posibilidad de combinarlo con bases de datos. Los *script* de PHP suelen intercalarse en el código HTML de una página, y el encargado de procesar esos *scripts* es el intérprete del servidor web.

El fin de un *script* PHP es generar una salida (que puede ser texto o código HTML, entre otras cosas), que puede ser diferente según el caso, ya que el objetivo de incluir scripts PHP en una página es obtener resultados dinámicos.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft

Windows, Mac OS X, RISC OS y probablemente alguno más.

PHP soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

De modo que, con PHP tiene la libertad de elegir el sistema operativo y el servidor de su gusto. También tiene la posibilidad de usar programación procedimental o programación orientada a objetos. Aunque no todas las características estándar de la programación orientada a objetos están implementadas en la versión actual de PHP, muchas bibliotecas y aplicaciones grandes (incluyendo la biblioteca PEAR) están escritas íntegramente usando programación orientada a objetos.

Con PHP no se encuentra limitado a resultados en HTML. Entre las habilidades de PHP se incluyen:
creación de imágenes, archivos PDF y películas Flash (usando libswf y Ming) sobre la marcha.

También puede presentar otros resultados, como XHTML y archivos XML. PHP puede autogenerar estos archivos y almacenarlos en el sistema de archivos en vez de presentarlos en la pantalla.

Quizás la característica más potente y destacable de PHP es su soporte para una gran cantidad de bases de datos. Escribir un interfaz vía web para una base de datos es una tarea simple con PHP. Las siguientes bases de datos están soportadas actualmente:

- Adabas D
- Ingres
- Oracle (OCI7 and OCI8)
- dBase
- InterBase
- Ovrimos
- Empress
- FrontBase
- PostgreSQL
- FilePro (readonly)
- mSQL
- Solid
- Hyperwave
- Direct
- MS-SQL
- Sybase
- IBM DB2
- MySQL
- Velocis
- Informix
- ODBC

- Unix dbm

También contamos con una extensión DBX de abstracción de base de datos que permite usar de forma transparente cualquier base de datos soportada por la extensión.

Adicionalmente, PHP soporta ODBC (el Estándar Abierto de Conexión con Bases de Datos), así que puede conectarse a cualquier base de datos que soporte tal estándar.

PHP también cuenta con soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. También se pueden crear sockets puros. PHP soporta WDDX para el intercambio de datos entre lenguajes de programación en web. Y hablando de interconexión, PHP puede utilizar objetos Java de forma transparente como objetos PHP Y la extensión de CORBA puede ser utilizada para acceder a objetos remotos.

PHP tiene unas características muy útiles para el procesamiento de texto, desde expresiones regulares POSIX extendidas o tipo Perl hasta procesadores de documentos XML. Para procesar y acceder a documentos XML, soportamos los estándares SAX y DOM. Puede utilizar la extensión XSLT para transformar documentos XML.

Si usa PHP en el campo del comercio electrónico, encontrará muy útiles las funciones Cybercash, CyberMUT, VeriSign Payflow Pro y CCVS para sus programas de pago.

Para terminar, contamos con muchas otras extensiones muy interesantes, las funciones del motor de búsquedas mnoGoSearch, funciones para pasarelas de IRC, utilidades de compresión (gzip, bz2), conversión de calendarios, traducción.

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 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 97 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(próxima a salir la versión 6 de la que ya se conocen algunos detalles), que utiliza el motor Zend, desarrollado con mayor meditación para cubrir las necesidades actuales y solucionar algunos inconvenientes de la anterior versión. Algunas mejoras de esta nueva versión son su rapidez - gracias a que primero se

compila y luego se ejecuta, mientras
que antes se ejecutaba mientras se interpretaba el código-, su mayor independencia del servidor
web -creando versiones de PHP nativas para más plataformas- y un API más elaborado y con más funciones.

En el último año, el número de servidores que utilizan PHP se ha disparado, logrando situarse cerca de los 5 millones de sitios y 800.000 direcciones IP, lo que le 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.

Tareas principales del PHP

Poco a poco el PHP se va convirtiendo en un lenguaje que nos permite hacer de todo. En un principio diseñado para realizar poco más que un contador y un libro de visitas, PHP ha experimentado en poco tiempo una verdadera revolución y, a partir de sus funciones, en estos momentos se pueden realizar una multitud de tareas útiles para el desarrollo del web.

Funciones de correo electrónico

Podemos con una facilidad asombrosa enviar 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...
Otras funciones menos frecuentes pero de indudable utilidad para gestionar correos electrónicos son incluidas en su librería.

Gestión de bases de datos

Resulta difícil concebir un sitio actual, 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 podremos editar el contenido de nuestro sitio con absoluta sencillez.

Gestión de archivos

Crear, borrar, mover, modificar...cualquier tipo de operación más o menos razonable que se nos pueda ocurrir puede ser realizada a partir de una amplia librería de funciones para la gestión de archivos por PHP. También podemos transferir archivos por FTP a partir de sentencias en nuestro código, protocolo para el cual PHP ha previsto también gran cantidad de funciones.

Tratamiento de imágenes

Evidentemente resulta mucho más sencillo utilizar Photoshop para una el tratamiento

de imágenes pero...¿Y si tenemos que tratar miles de imágenes enviadas por nuestros internautas?

La verdad es que puede resultar muy tedioso 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 utilizamos el mismo diseño y solo cambiamos el texto. Podremos por ejemplo crear un botón haciendo una única

llamada a una función en la que introducimos el estilo del botón y el texto a introducir obteniendo automáticamente el botón deseado.

A partir de la librería de funciones graficas podemos hacer esto y mucho más.

Muchas otras funciones pensadas para Internet (tratamiento de cookies, accesos restringidos, comercio electrónico...) 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 uno va creando 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 nos presentan son sorprendentemente vastas. Lo único que se necesita es un poco de ganas de aprender y algo de paciencia en nuestros primeros pasos. El resultado puede ser muy satisfactorio.

Configuración del comportamiento de PHP

En este punto se desarrollaran brevemente las posibilidades de configuración que nos ofrece PHP, para el uso que le daremos en el curso, usaremos por lo general la configuración que viene sugerida en la instalación de XAMPP, la cual abordaremos más adelante en esta misma unidad.

El archivo de configuración php.ini

El archivo de configuración (llamado php3.ini en PHP 3, y simplemente php.ini a partir de PHP 4) es leído cuando arranca PHP.
El archivo php.ini es el que contiene nuestra configuración de PHP, con el que podemos controlar
muchos aspectos de su funcionamiento. En estas páginas intentaremos explicar para que sirve cada una de sus instrucciones y cuál es la mejor forma de configurarlo. La sistemática de las páginas consiste en seguir el mismo orden interior del php.ini, aunque puede que haya ligeras diferencias con sus tu copia, debidas a pequeños cambios entre versiones.

¿Qué es el archivo php.ini?

Este archivo sirve para indicar una serie de valores que determinan el comportamiento del intérprete PHP. Lo encontramos dentro de la distribución

php en el directorio raíz.

Se trata de un archivo de texto, que podemos abrir con cualquier editor que trabaje con texto simple
(*.txt).

Lo primero que debemos hacer es en efecto editar una de las dos versiones disponibles, configurarla de acuerdo a nuestras necesidades, y guardarla con el nombre php.ini.

¿Cual escoger? las dos son el mismo archivo, con distintos valores por defecto. El texto marcado con corchetes, como [PHP] indica una cabecera de sección.

Las instrucciones se llaman directivas, y están formadas por una pareja compuesta por la clave y su valor, por ejemplo: asp_tags = Off.

Diferencian entre mayúsculas y minúsculas. No es lo mismo asp_tags que Asp_tags.

También verán que algunas directivas comienzan con ; lo que quiere decir que están comentadas ya que no son necesarias por defecto. Se deben desactivar si se necesitan esas funcionalidades.

¿Cómo trabaja el archivo php.ini?

Si tenemos PHP como módulo del servidor, el archivo php.ini se lee cada vez que se reinicia. Por lo tanto tienen que reiniciar para que actualice los cambios.

Las directivas

Veremos a continuación cada una de las directivas y su significado, siguiendo el orden que podriamos ver en nuestro php.ini. Muchas directivas vienen con valores por defecto, o sin valor determinado, o comentadas (inactivas). Una buena política es dejarlas como están, salvo que sepas exactamente que estás haciendo.

Los valores que indicamos en esta página son indicativos. Lo que pretendemos es explicar el valor de cada directiva (al menos las que conocemos), no proponer un php.ini modélico.

Opciones de lenguaje

En esta primera sección encontramos algunas instrucciones generales sobre el funcionamiento de
PHP:

engine = On

activa la interpretación de scripts php (si php está cargado como módulo de apache). Esta directiva, en unión de httpd.conf, permite habilitar o deshabilitar php en directorios determinados.

short_open_tag = On

Permite usar en tus scripts etiquetas php abreviadas <? ... ?>, y el atajo para imprimir variables <%=
$valor %>. Si el valor es off, deberán usar la forma <?php ... ?>

o <script>. Se recomienda ponerlo a off para mayor portabilidad del código

asp_tags = Off
Permite usar etiquetas estilo asp <% ... %>. Deshabilitado por defecto precisión = 14 número máximo de decimales visualizados y 2k_compliance = On Forzar compatibilidad con el año 2000.

output_buffering = Off
Permite enviar cabeceras http (cookies por ejemplo) desde puntos distintos al inicio del script. Además de valores on | off puedes fijar aquí el tamaño máximo (en bytes) de las líneas http permitidas, por ejemplo: output_buffering = 4096
Puedes deshabilitar esta función con carácter general aquí, y habilitarla en partes concretas de tus scripts utilizando las funciones de buffer correspondientes (por ejemplo ob_start()).

Cuando output buffering esta activado, PHP no lanza las cabeceras HTTP al inicio de la ejecución del script, sino que las almacena temporalmente en un buffer de memoria, lo que te permitirá modificar o añadir instrucciones HTTP durante la ejecución del script, que se enviarán solo cuando este
finalice.

Esta posibilidad está penalizada por una disminución del rendimiento.

output_handler =
Con esta directiva puedes redirigir toda la salida de tus scripts a una función PHP. Es preferible no habilitar esta opción y establecerla si es preciso en cada uno de tus scripts.

zlib.output_compression = Off
habilita la librería zlib de forma que los datos de salida del script se envían comprimidos. Puedes indicar valores off|on o precisar el tamaño del buffer (por defecto es de 4 KB).

;zlib.output_handler =
Si tienes activada la opción anterior, no puedes usar la directiva output_handler; con similar funcionalidad tienes zlib.output_handler.

implicit_flush = Off
Intenta enviar al cliente el contenido de la memoria intermedia de salida. O dicho coloquialmente,
"envía lo que tengas hasta ahora, en lugar de esperar a completarlo". Equivale a llamar la función flush() despues de cada llamada echo o print y cada segmento html. Es desaconsejable su activación, siendo preferido usar la función flush() cuando sea necesario.

unserialize_callback_func=
relacionado con las funciones serialize(). Francamente no se mas sobre el tema.

allow_call_time_pass_reference = Off
Tradicionalmente podías construir una función y al usarla, decidir si pasabas o no el valor de una variable por referencia (&$var). Ahora esto es desaconsejado y se recomienda especificar que los valores serán pasados por referencia en la propia

declaración de la función (function blah (&$var))

safe_mode = Off
Para activar el modo seguro de PHP.
Si usas PHP como CGI, "debes" activar safe_mode y especificar el valor de safe_mode_exec_dir, con lo cual aseguras que el usuario solo pueda acceder a la información existente en las carpetas especificadas.

safe_mode_gid = Off
Por defecto, con safe_mode On PHP hace un chequeo UID del fichero al abrirlo. Con esta directiva
puedes especificar en su lugar un chequeo GID

safe_mode_include_dir =
Los archivos que estén en este directorio podrán ser utilizados con include/require en safe_mode On sin necesidad de chequeos UID/GID

safe_mode_exec_dir =
Si el PHP se utiliza en modo seguro, la función system() y el resto de funciones que ejecutan programas del sistema solo actuaran sobre archivos ejecutables que estén en el directorio indicado.

safe_mode_allowed_env_vars = PHP_ Puedes proporcionar aquí una serie de prefijos (separados por ;). Si indicas estos prefijos, en safe_mode los usuarios solo podrán alterar variables de entorno cuyo nombre comience con ese prefijo. Si esta directiva esta vacía, en safe_mode podrán modificarse todas las variables de entorno.

safe_mode_protected_env_vars = LD_LIBRARY_PATH una lista de variables de entorno (separadas por ;) que no pueden variarse via putenv() incluso aunque safe_mode_allowed_env_vars lo permita open_basedir = Limita los archivos que se pueden abrir por PHP al árbol de directorios especificado. Cuando un script intenta abrir un archivo con, por ejemplo, fopen, se comprueba su localización. Si el fichero está fuera del árbol de directorios especificado, PHP se negará a abrirlo. Todos los enlaces simbólicos son resueltos, de modo que no es posible evitar esta limitación usando uno de ellos.
El valor especial indica que el directorio base será aquel en el que reside el script.

Bajo Windows, los directorios se separan mediante punto y coma. En el resto de sistemas, con dos puntos ":". Como módulo de Apache, los senderos para open_basedir de los directorios padre se heredan ahora automáticamente.

El valor por defecto es permitir abrir todos

los archivos. Esta directiva es independiente

de Safe Mode.

disable_functions =
Con esta directiva puedes inhabilitar con carácter general determinadas funciones PHP. Basta con incluirlas separadas por punto y coma (";"). Al igual que la anterior, es independiente de Safe Mode. highlight... permite especificar los colores a utilizar por el coloreador de sintaxis interno de PHP expose_php = On Permite controlar si PHP debe o no revelar su presencia en el servidor, por ejemplo incluyéndose en las cabeceras http

del servidor.

Limites al empleo de recursos

max_execution_time = 30
Fija el tiempo máximo en segundos que se le permite usar a un script antes de ser finalizado por el intérprete. Así se evita que scripts mal escritos puedan bloquear el servidor.

max_input_time = 60
Tiempo máximo en segundos que el script puede invertir en analizar datos recibidos.

memory_limit = 8M
Fija el tamaño máximo de memoria en bytes que se permite reclamar a un script. Así se evita que script mal escritos se coman toda la memoria disponible de un servidor.

Gestión y archivo de errores

error_reporting = E_ALL
Fija el nivel (detalle) con el que PHP te informa de errores.

Esta directiva vuelca el informe de errores en la pantalla, y su uso está desaconsejado en páginas en producción, ya que el error puede revelar información sensible. Lo recomendado es permitir mostrar errores, con el máximo detalle posible, mientras desarrollas el script PHP; y cuando está terminado y en producción, deshabilitar el mostrado de errores en pantalla y activar en su lugar el archivo de errores.

Como cada nivel de informe de error está representado por un número, puedes designar el nivel deseado sumando valores:

1 errores normales
2 avisos normales
4 errores del parser (error de sintaxis)
8 avisos de estilo no críticos

El valor por defecto para esta directiva es 7 (se muestran los errores normales, avisos normales y errores de parser).

También puedes designar el nivel de error

nominativamente: Algunas combinaciones son:
error_reporting = E_ALL & ~E_NOTICE
muestra todos los errores críticos, excluyendo advertencias que pueden indicar mal funcionamiento del código pero no impiden la ejecución del intérprete.

error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR
muestra solo errores.

error_reporting = E_ALL
muestra todos los errores y advertencias.

display_errors = Off

determina si los errores se visualizan en pantalla como parte de la salida en HTML o no.

Como queda dicho, es desaconsejado mostrar errores en pantalla en páginas visibles al público.

display_startup_errors = Off
Incluso con display_errors on, por defecto PHP no muestra los errores que detecta en la secuencia de encendido. Con esta directiva puedes mostrar estos errores. Desaconsejado activarla.

log_errors = On
Guarda los mensajes de error en un archivo. Normalmente el registro del servidor. Esta opción, por tanto, es específica del mismo.

log_errors_max_len = 1024
Especifica el tamaño del archivo error_log. Si tiene un valor 0 significa que no hay restricción de
tamaño

ignore_repeated_errors = Off
Si está activado, no archiva mensajes repetidos. No se consideran mensajes repetidos aquellos que no provienen de la misma línea.

ignore_repeated_source = Off
Si está activado, considera repetidos los mensajes de error iguales, aunque provengan de distinta línea / script

report_memleaks = On
Mostrar o no. memory leak se refiere a cuando (por error) el script no libera la memoria usada cuando ya no la necesita, y en consecuencia usa cada vez mas hasta llegar a agotarla.

track_errors = Off
Si lo activamos, tendremos el último mensaje de error/advertencia almacenado en la variable
$php_errormsg

html_errors = Off
Si activo, no incluye etiquetas HTML en los mensajes de error.

docref_root = /phpmanual/ y docref_ext = .html
Si tienes html_errors activado, PHP automáticamente incluye enlaces en el mensaje de error que te dirigen a la página del manual que explica la función implicada. Puedes bajarte una copia del manual y indicar su ubicación (y extensión del archivo) usando estas directivas.

error_prepend_string = ""
Cadena a añadir antes de cada mensaje de error.

error_append_string = ""
cadena a añadir después del mensaje de error.

;error_log = filename
Nombre del fichero para registrar los errores de un script. Si se utiliza el valor especial syslog, los errores se envían al registro de errores del sistema.
Como verán, esta comentado (inhabilitado) por defecto.

Gestión de datos

track_vars
Esta directiva crea arrays $HTTP_GET_VARS, $HTTP_POST_VARS y $HTTP_COOKIE_VARS con los datos introducidos con los métodos GET, POST y con cookies. Desde PHP 4.0.3 está siempre activada.

;arg_separator.output = "&"
El carácter que se empleará en las urls generadas por PHP para separar argumentos (valores pasados
via url). & es el separador por defecto.

;arg_separator.input = ";&"
separadores que usará PHP cuando analice una url suministrada para almacenarla en variables variables_order = "GPCS" Esta directiva fija el orden (precedencia) en que PHP registrará y interpretará las variables de entorno (de izquierda a derecha en el orden indicado). Los valores posibles se toman con las iniciales del método usado para asignar el valor a la variable: Get, Post, Cookie, Enviroment y Server. Fijando por ejempo el valor a "GP", hará que el PHP ignore por completo las cookies y que sobreescriba las variables recibidas por GET con las que tengan el mismo nombre y vengan por POST.

En php.ini encontrarás una directiva semejante en desuso (no recomendada) que es gpc_order

register_globals = Off
Permite registrar automáticamente (o no) las variables EGPCS como globales. Por razones de seguridad se recomienda desactivar el registro.

register_argc_argv = Off
Esta directiva instruye a PHP si debe declarar las variables argv&argc (arrays predefinidos que almacenan los parámetros pasados (argv) y su numero (argc).

post_max_size = 8M
Tamaño máximo de los datos que PHP aceptará por el método POST

Magic quotes

magic_quotes_gpc = Off
Fija el estado magic_quotes para operaciones GPC (Get/Post/Cookie). Si magic_quotes vale on, todas las ' (comilla sencilla), " (comilla doble), (barra invertida) y los NUL son automáticamente marcados con una barra invertida. Si además magic_quotes_sybase vale on, la comilla sencilla es marcada con otra comilla sencilla en lugar de la barra invertida.

magic_quotes_runtime = Off
Si se habilita magic_quotes_runtime, muchas de las funciones que devuelven datos de algún tipo de fuente externa incluyendo bases de datos y archivos de texto devolverán

las comillas marcadas con una barra invertida. Si también está activo magic_quotes_sybase, la comilla simple es marcada con una comilla simple en lugar de la barra invertida.

magic_quotes_sybase = Off Si magic_quotes_sybase está a on, la comilla simple es marcada con una comilla simple en lugar de la barra invertida cuando están habilitados magic_quotes_gpc o magic_quotes_runtime.

Más directivas de Gestión de datos

auto_prepend_file = y auto_append_file =
permiten indicar la ruta y nombre de un archivo que se añadirán antes o despues (respectivamente)
de todos los archivos php que se ejecuten.
El valor especial none desconecta la adición automática de archivos.
Si el script es terminado con exit(), no tendrá lugar la adición automática señalada con auto_append_file.

Los archivos indicados con estas directivas se incluirán como si fuesen llamados mediante la función include(), así que se utiliza include_path.

;default_charset = "iso-8859-1"
Por defecto, el código de caracteres indicado por PHP en la cabecera de salida.

default_mimetype = "text/html"
Por defecto, el tipo mime de salida de datos. Cada MIMETYPE define el formato de los datos (por ejemplo, texto/hmtl, jpg, gif ....)

;always_populate_raw_post_data = On
PHP crea la variable $HTTP_RAW_POST_DATA cuando recibe datos via POST cuyo tipo MIME no reconoce (almacena los datos en esta variable sin analizarlos). Con esta directiva se ordena que se cree siempre la variable $HTTP_RAW_POST_DATA, aunque el tipo MIME sea conocido.

;allow_webdav_methods = On Permite manejar las peticiones http propias de webdav.

Rutas y directorios

include_path = ".;c:phpincludes"
Permite especificar una lista de directorios en los que las funciones require(), include() y fopen_with_path() buscaran los archivos requeridos.

El formato es similar a la variable de entorno de sistema PATH: una lista de directories separados por dos puntos en UNIX o por punto y coma en Windows. Ejemplo unix seria include_path=.:/home/httpd/php-lib y en Windows

include_path=".;c:wwwphplib".
El valor por defecto para esta directiva es . (sólo el directorio actual).

doc_root =
Indica el "Directorio raiz" donde estan nuestras paginas php en el servidor.
Sólo se usa si no está vacío. Si PHP se configura con safe mode, no se interpretaran las páginas php situadas fuera de este directorio. Ojo con los servidores virtuales que apuntan a zonas distintas del servidor.

user_dir =
El directorio raíz para los archivos PHP bajo el directorio inicial de un usuario (/~usuario). Normalmente se deja vacio .

extension_dir = ./
En qué directorio debe buscar el PHP las extensiones dinámicas a cargar. Bajo Windows, por defecto si no pones ningún valor en esta directiva, se buscaran en c:php4extensions.

enable_dl = On
Esta directiva sólo es útil en la versión del PHP como módulo del Apache. Puede habilitar o deshabilitar para un servidor virtual o para un directorio la carga dinámica de extensiones de PHP mediante dl().

La razón principal para desactivar la carga dinámica es la seguridad. Con la carga dinámica es posible ignorar las restricciones para abrir archivos establecidas con open_basedir.

El valor por defecto es permitir la carga dinámica, excepto cuando se usa safe_mode. En modo seguro, es imposible usar dl().

cgi.force_redirect = 1
Por defecto se activa. Es una directiva importante de seguridad que "debes" activar si ejecutas en tu
apache PHP como cgi (no es necesaria si tienes PHP como modulo, o si usas como servidor el IIS de microsoft).

; cgi.redirect_status_env = ;
En conjunción con cgi.force_redirect y servidores distintos de Apache o iPlanet.

; fastcgi.impersonate = 1;
En conjunción con IIS y FastCGI

Subir ficheros

file_uploads = On
Permitir o no subir (upload) ficheros vía HTTP.

upload_tmp_dir =
Carpeta o directorio utilizable para guardar temporalmente archivos subidos por PHP. Si no se especifica, usará el designado por defecto por el servidor. El usuario que esté ejecutando el script debe tener permiso de escritura en ese directorio.

upload_max_filesize = 2M
Tamaño máximo de archivos que pueden subirse.

Directivas relacionadas con fopen

allow_url_fopen = On Permite pasar urls (http, ftp) a la función fopen(), en lugar de la ubicacion fisica del archivo

;from="john@doe.com" define el email a usar como contraseña para ftp anónimo

;user_agent="PHP" define la "firma" que dejará PHP en el servidor remoto de donde coge los archivos default_socket_timeout = 60 timeout en segundos para la apertura de sockets

; auto_detect_line_endings = Off
Si activo, PHP detectara automáticamente el carácter que indica fin de linea (distinto en windows,
linux y windows)

Extensiones dinámicas

extensión= Qué extensiones dinámicas debe cargar el PHP cuando arranca. Debes elegir el archivo que corresponde a tu sistema operativo: por ejemplo extension=msql.dll para windows, extension=msql.so para linux.

Ojo, aqui solo indicamos la extensión de los archivos, no su ubicación. Los archivos DEBEN estar en el directorio especificado más arriba con extension_dir.

Las versiones más recientes de PHP traen "de serie" los módulos MYSQL, ODBC y GD por lo que NO
tienes que cargar sus extensiones.

Configuración de módulos de PHP

define_syslog_variables = Off
Permite definir variables del sistema. Recomendado Off.

;browscap = extra/browscap.ini
El archivo browscap.ini es un archivo de texto que contiene información sobre las cadenas de identificación que usa cada navegador.

Mediante esta directiva indicas a PHP donde tienes browscap.ini; se usa conjuntamente con la función get_browser().

Directivas de Configuración de Correo

Si usas PHP bajo linux, puedes enviar correo usando tu propio PC con sendmail; con windows no tienes esa posibilidad, por lo que para enviar correos desde un script PHP con la funcion mail() tienes que delegar en tu configuración de correo ordinaria, la que usas por ejemplo con outlook para enviar y recibir correo.

Este sería un ejemplo bajo windows:

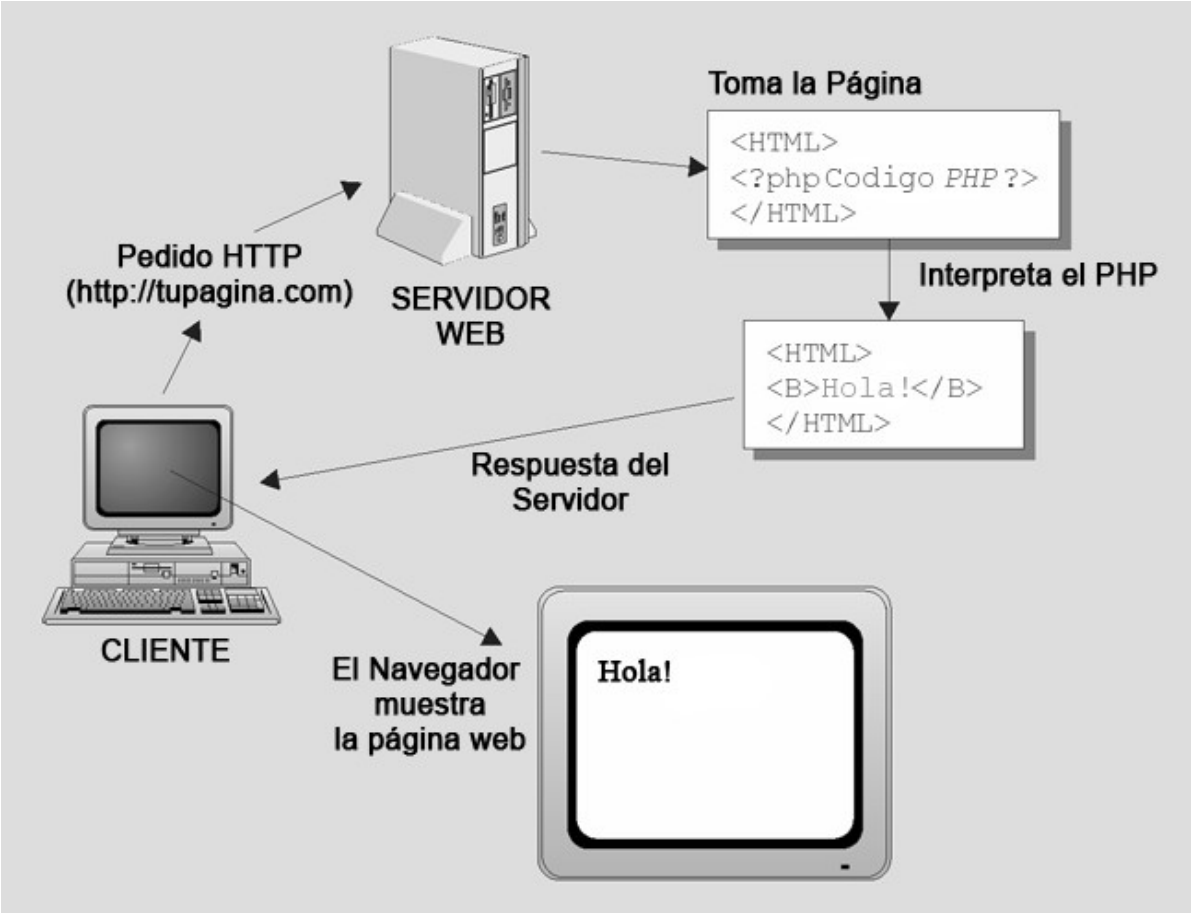
SMTP = mailhost@teleline.es
Este sería el caso si tu conexión a internet te la proporciona telefónica. Especificamos la dirección del servidor smtp (correo saliente).

sendmail_from= webmaster@misitio.com
La dirección del remitente ("De:") para los correos enviados desde PHP bajo Windows.

Funcionamiento.

PHP es un **lenguaje de lado de servidor**. ¿Qué significa esto? Que lo que el programador codifique en PHP dentro de una página de Internet, no va a ser interpretado por el cliente (navegador), si no por un servidor que procese el código y devuelva al cliente el código HTML resultante.

Cuando nosotros le pedimos al navegador que nos lleve a una página en Internet, el navegador le pide al servidor el código de esta página, lo interpreta y lo muestra en pantalla. Si en el código de la página a la que queremos acceder hay *scripts* PHP (y por supuesto, el documento tiene **extensión .php**), antes de que el servidor le entregue al cliente el código de la página, estos *scripts* son interpretados por el servidor, y en vez de devolverle código PHP al cliente, le devuelve el código HTML resultante de procesar esos *scripts* PHP.



Instalación

Para poder ver los resultados de los *scripts* PHP, es necesario contar con un servidor que pueda interpretarlos.

Un servidor que permite esto es el **Apache**. Cuando uno quiere hacer un sitio web con PHP, al momento de ponerlo online debe asegurarse que el hosting a contratar cuenta con soporte para PHP y corre el servidor Apache.

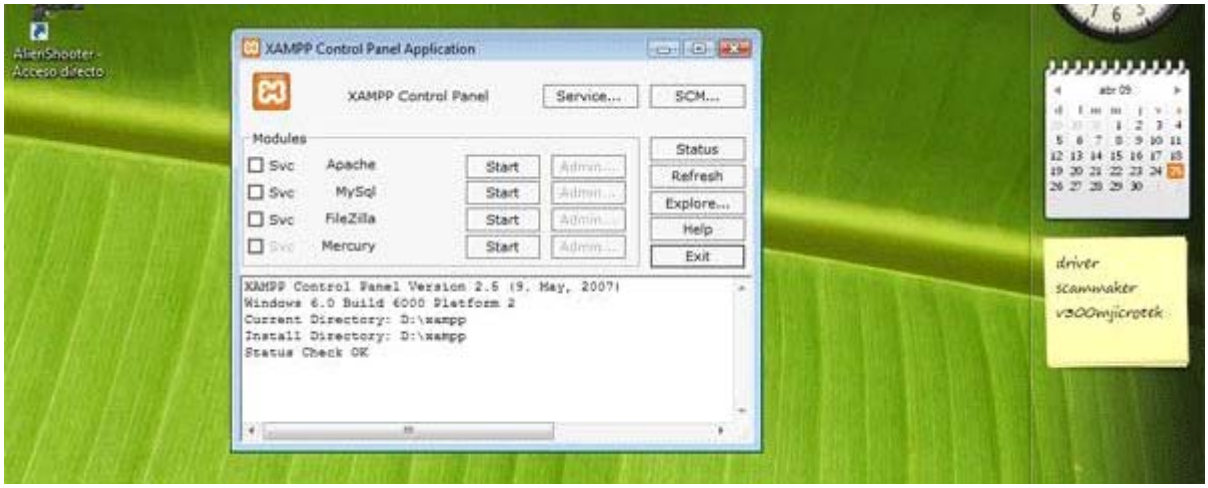
Si se van a usar bases de datos (generalmente **MySQL**) también hay que asegurarse de que el hosting lo soporte.

Para trabajar localmente sin depender de estar subiendo constantemente los archivos al hosting para poder verlos en funcionamiento, podemos instalar el servidor Apache y MySQL en nuestra computadora. Un programa que facilita esta instalación es el XAMPP, es una forma fácil de instalar la distribución Apache que contiene MySQL, PHP y Perl. XAMPP es realmente simple de instalar y usar - basta descargarlo, extraerlo y comenzar.

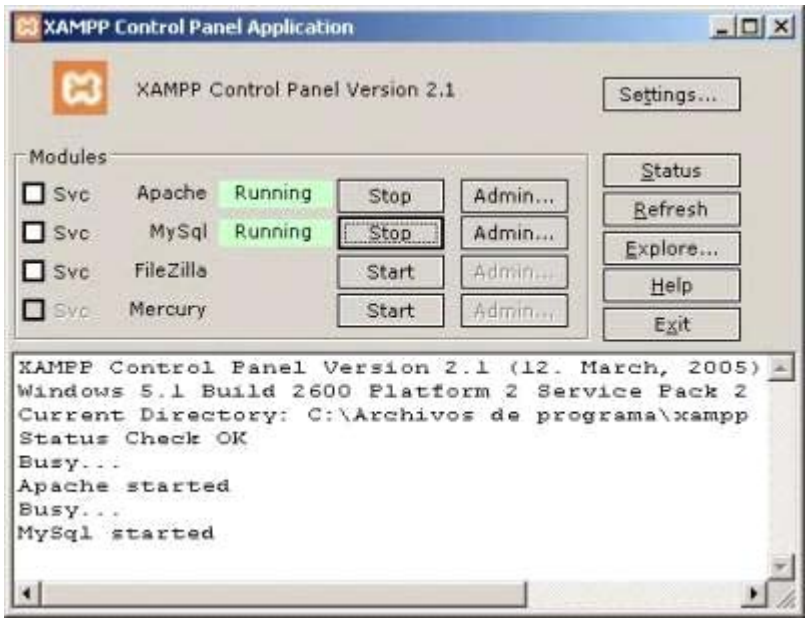
Puede descargarse de http://www.apachefriends.org/es/xampp.html

(ya lo instalamos en la unidad anterior)

Una vez concluida la instalación, debemos ejecutar el programa, luego de hacerlo, deberíamos ver el panel de control que se muestra aquí a continuación:

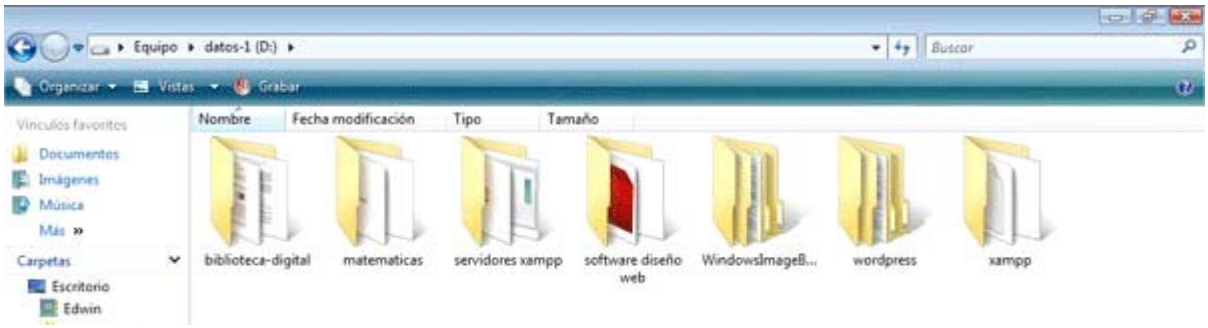


Lo normal es activar los dos primeros, para ello hacemos click en los botones que dicen “Start” pertenecientes a Apache y MySQL, debería aparecer la palabra “Running” al costado de cada uno de ellos, como podemos ver a continuación:

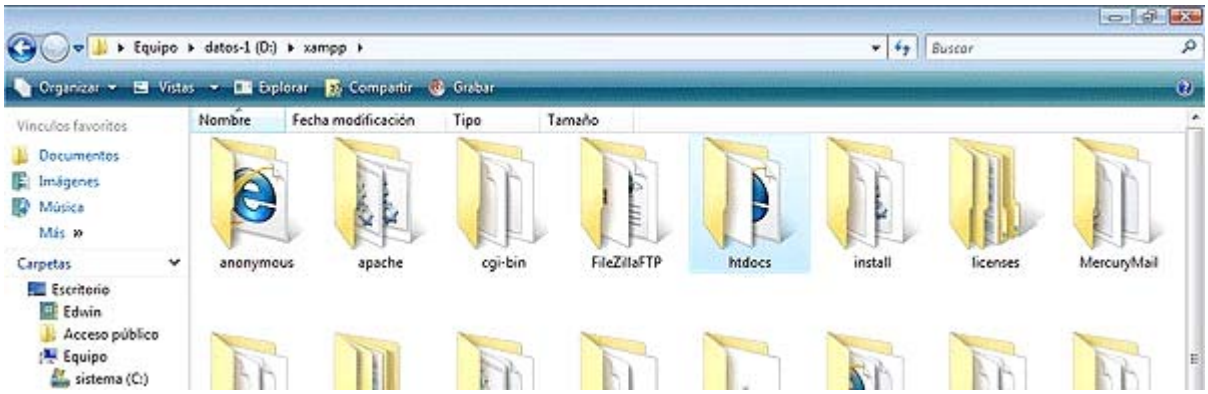


Habilitar estas dos opciones nos permitirá que desde cualquier navegador tecleemos la IP local (http://127.0.0.1, o bien http://localhost) y accedamos a la verdadera interfaz de la aplicación, desde la cual tendremos acceso a nuestras bases de datos MySQL a través del phpMyAdmin (el cual veremos más adelante), al módulo de estadísticas Webalizer, o a la configuración de los servidores FTP y de correo.

Tendremos entonces en el disco que hayamos seleccionado una carpeta llamada XAMPP, si es que usamos el nombre de directorio que viene por defecto, o el que hayamos indicado al momento de la instalación.



Dentro de la carpeta del programa, encontraremos la carpeta “htdocs” Es el sitio que el server Apache del Xampp “publica” como sitios web, es decir que toda carpeta que esté dentro y contenga archivos html, php u otros formatos son leídos en línea por el navegador y mostrados como una página web.



El establecimiento de la seguridad es otro de sus puntos destacables, XAMPP es sin duda, toda una joya que está disponible tanto en versión Windows como en Linux.

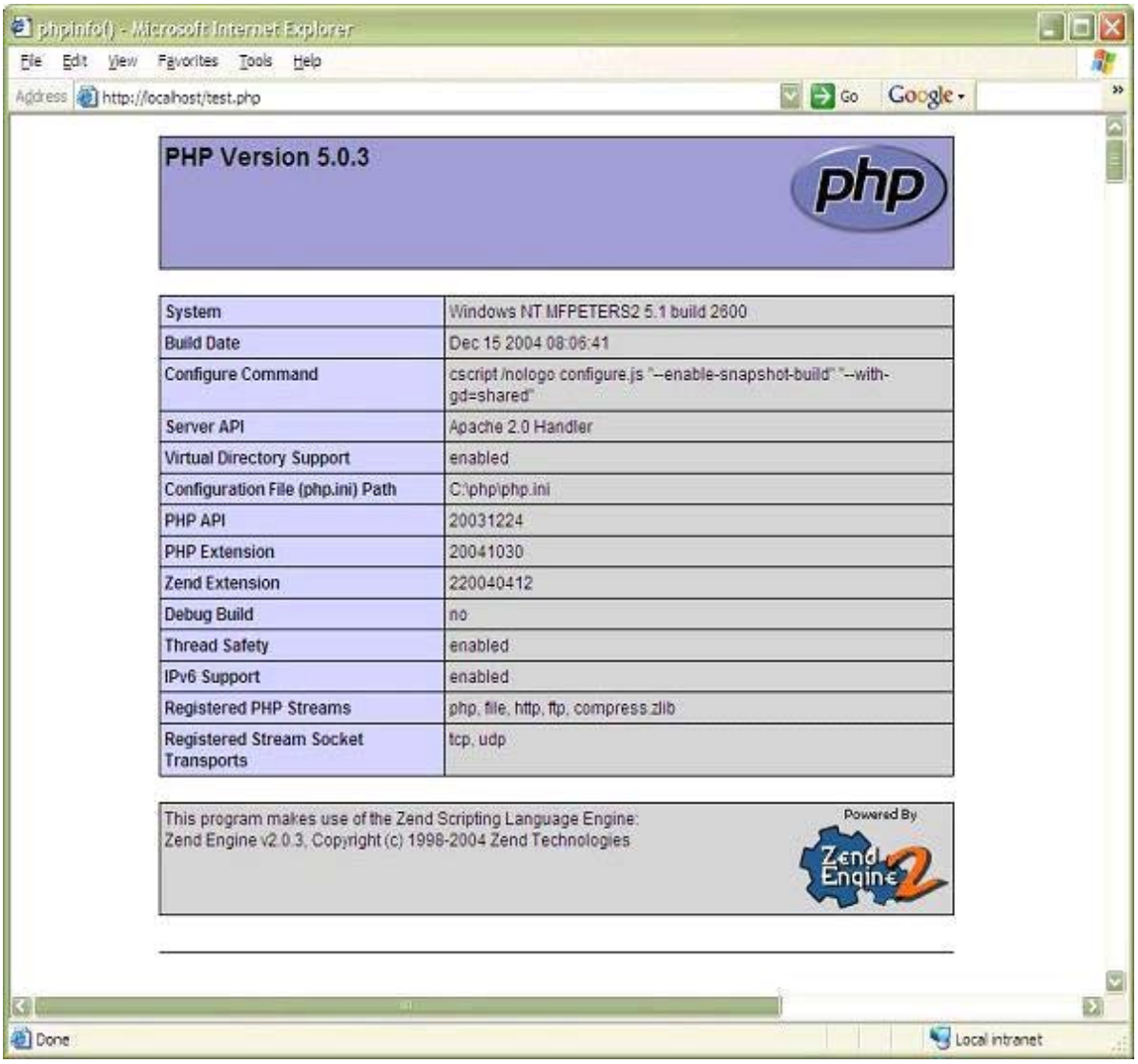
Incluso existe una beta para Solaris/SPARC, mientras que los usuarios de Mac tienen una buena alternativa en MAMP. Nosotros no la modificaremos y utilizaremos sus valores por defecto, para usuarios “root” y para contraseñas vacío, es decir “” (que quedará claro más adelante cuando veamos ejemplos que requieran esta identificación.)

Lo que nos quedaría ahora es ver la página web que nos muestra cuándo ingresamos en http://127.0.0.1, o bien http://localhost, desde allí podremos ver algunos ejemplos que trae instalado el programa y los módulos del mismo, a continuación la mostramos:

En definitiva, si queremos ver un archivo de PHP, solo debemos copiarlo entonces en el directorio htdocs y ejecutarlo desde el navegador. Por ejemplo, si quisiéramos ver el archivo llamado “phpinfo.php”, luego de copiarlo en dicho directorio, abrimos nuestro navegador y escribimos lo siguiente:

http://127.0.0.1/phpinfo.php o http://localhost/phpinfo.php y le damos enter

Deberíamos obtener como respuesta una página como la siguiente:



¿Qué podemos hacer con PHP?

Usando PHP en un sitio web, se puede lograr un mayor dinamismo del contenido y, sobre todo combinándolo con la gestión de bases de datos, lograr hacer funcionar algunos de los siguientes ejemplos:

- Foros
- Guestbooks
- Blogs
- Calendarios
- Sistemas de Carrito de Compras
- Restricción de acceso a sitios web mediante user/password
- Registro de estadísticas
- Proceso de Formularios de Email

Intercalando PHP con HTML

Un documento PHP puede tener íntegramente código PHP así como intercalaciones de código PHP y
HTML.

Muchas veces, el contenido de una página no es fijo. Ejemplos de esto hay muchísimos, desde algo trivial como mostrar la hora actual en el mensaje de bienvenida de un sitio web, hasta mostrar una lista de productos que un visitante del sitio haya ido agregando a un carrito de compras, los cuales se van listando a medida que el visitante los selecciona. En estos casos, el HTML “nos queda corto” y necesitamos usar scripts PHP para generar o administrar un contenido generado dinámicamente, el cual seguramente será sólo una porción de la página y no toda entera. Entonces, el contenido estático de la página estará *codificado* en HTML (diseño, estructura, textos fijos, etc), y e contenido dinámico estará *programado* en PHP.

Para comenzar a introducirnos en la sintaxis de PHP y en cómo intercalar PHP con HTML, se hará uso de ejemplos que no reflejan el “dinamismo” que se puede obtener con PHP, pero ayudarán a dar el primer paso a tomar una idea de cómo se puede llegar a eso.

Tenemos el siguiente documento holamundo.php, con HTML y PHP intercalados:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Hola mundo</title>
</head>
<body>
<?php
echo "Hola mundo";
?>
```

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

Dentro del body de esa página, nos encontramos con la sentencia echo "Hola mundo"; que muestra por pantalla la frase "Hola mundo". Sería exactamente lo mismo que figure esa sentencia PHP allí a que sólo figuren las palabras "Hola mundo", que sería un texto común de HTML.

- *echo:* visualiza una o más cadenas. Realizará un volcado en la pantalla del contenido indicado. Usamos comillas cuando queremos que PHP trate literalmente el contenido (que vuelque una cadena de caracteres sin manipularla) y paréntesis cuando queremos que PHP calcule el contenido antes de volcarlo.

Los elementos que encontramos en este *script* son, por un lado, los **delimitadores** de PHP y por otro la **sentencia**. Los delimitadores son **<?php** y **?>**, y sirven para que cuando el servidor reciba la petición de holamundo.php, sepa cuándo tiene que procesar código PHP y cuándo debe dejar de hacerlo. Al encontrarse con el delimitador de apertura **<?php**, el servidor sabe que allí tendrá que empezar a interpretar el código PHP para "convertirlo" en su salida correspondiente en HTML, y al encontrarse con el delimitador de cierre **?>**, sabe que debe dejar de interpretar PHP. De esta manera, usando los delimitadores de código PHP, podemos intercalar libremente *scripts* PHP con código HTML.

Por otro lado, tenemos la sentencia PHP que el servidor interpretará. **echo** es una instrucción del lenguaje PHP que toma como argumento un string (cadena de caracteres) que aparezca a su derecha, y lo muestra por pantalla.

En realidad, a ese argumento que recibe lo imprime en el código HTML, en el lugar donde figura el código PHP que lo ejecuta. Si nosotros ejecutamos la página holamundo.php en un navegador y después miramos su código fuente desde el mismo, el resultado será este:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Hola mundo</title>
</head>
<body>
Hola mundo
</body>
</html>
```

Se verá que el servidor interpretó el *script* PHP y en su lugar "imprimió" su salida correspondiente,
que en este caso era simplemente las palabras "Hola mundo".

Volviendo a la sentencia **echo "Hola mundo";** hay un par de cosas más para notar. La frase "Hola mundo" está entre **dobles comillas** por ser un **string o cadena de caracteres**.

También puede ir entre comillas simples, aunque guardan diferencias que se verán más adelante. El otro punto muy importante es el **punto y coma** (**;**).

Todas las sentencias de PHP terminan con un punto y coma. Es la manera que tiene el intérprete de saber que el programador allí quiso terminar una sentencia.

Mientras terminemos todas las sentencias con un punto y coma, no es siquiera necesario que estén separadas por renglones, podrían estar una detrás de la otra en el mismo renglón, y por otro lado, tampoco es necesario que una misma sentencia la expresemos en un solo renglón, podría estar repartida en varios que hasta que no haya un punto y coma en el código, el intérprete de PHP sabe que todavía no se terminó la sentencia.

NUNCA debemos olvidar poner un punto y coma al final de cada sentencia.

Si ha intentado usar este ejemplo, y no produjo ningún resultado, preguntando si deseaba descargar el archivo, o mostró todo el archivo como texto, lo más seguro es que PHP no se encuentra habilitado en su servidor.

Otros Ejemplos:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Hola mundo</title>
</head>
<body>
<?php
echo “Este
es
el clasico “; echo "Hola mundo";
?>
</body>
</html>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Hola mundo</title>
</head>
<body>
<?php
echo "Este es el clásico "; echo "Hola mundo";
```

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

Ambos códigos mostrarán por pantalla el mismo mensaje:

"Este es el clásico Hola Mundo"

Así como hasta ahora se ha usado texto con la instrucción echo, se puede pensar que si ese texto queda impreso en el código fuente de la página como HTML cuando ya fue procesado por el intérprete PHP del servidor y devuelto al navegador, también podríamos imprimir directamente código HTML.

Por ejemplo, en vez del *script* anteriormente usado, podríamos remplazarlo por algo como:

```
<?php
echo "<p>Hola mundo</p>";
?>
```

y si ejecutamos holamundo.php en el navegador y vemos su código fuente, nos encontraremos con:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Hola mundo</title>
</head>
<body>
<p>Hola mundo</p>
</body>
</html>
```

Sintaxis de PHP

Ya se adelantaron en la sección anterior un par de temas de sintaxis de PHP. Por un lado, las sentencias siempre finalizan con un punto y coma. Por otro, el código PHP siempre debe estar encerrado por sus delimitadores, **<?php** y **?>**.

Comentarios

Para que en futuras revisiones a tu código por tu parte y sobre todo por parte de otros programadores este código sea inteligible para la mente humana, es una buena costumbre comentar lo que se está haciendo en PHP pero en palabras.

De esta manera el código será mucho más sencillo de comprender y a su vez de modificar, corregir, ampliar... etc.

Los comentarios que soporta PHP son los de C, C++ y los del shell de Unix, así podemos usar // y /*
*/ para comentarios y comentarios multilínea respectivamente como haríamos en C:

```
<?php
echo 'Esto se ve';
// esto no se ve
echo 'esto también se ve';
/*
Esto tampoco
se ve
*/
?>
```

O también podemos usar # como en los comentarios del shell de Unix:

```
<?php
# esto no se ve
echo 'pero esto sí';
?>
```

Tendremos cuidado con no poner un comentario multilínea en el interior del otro:

```
<?php
/* /* No hacer nunca esto */ */
?>
```

PHP interpreta como comentario desde el primer /* al primer */ sin tener en cuenta que hay otro */, esto nos producirá un error así que habrá que tener cuidado.

phpinfo()

Como mencionáramos anteriormente, existe en PHP una función que nos permite visualizar los contenidos del archivo php.ini que antes mencionáramos.

Esta función Imprime una gran cantidad de información sobre el estado actual de PHP.

Esto incluye información sobre las opciones de compilación de PHP y sus extensiones, la versión de
PHP, información del servidor y el entorno (si ha sido compilado como módulo), el entorno de PHP,

información de la versión del SO, rutas, valores de configuración maestros y locales, cabeceras HTTP
y la licencia de PHP.

Dado que cada sistema es configurado de forma distinta, phpinfo() es usado con frecuencia para verificar los parámetros de configuración y las variables predefinidas disponibles en un sistema dado. Asimismo, phpinfo() es una valiosa herramienta de depuración ya que contiene todos los datos EGPCS (Entorno, GET, POST, Cookie, Servidor).

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