

Apache + PHP + MySql + PhpMyAdmin como módulo de Apache

Este tutorial esta basado en la instalación de Windows XP + Apache 2.0.55 + MySQL 5.0.19 + PHP 5.1.2 + PhpMyAdmin 2.8.0.3.

NOTA: Te recomiendo que antes de empezar a realizar este tutorial, si tu PC esta conectada a alguna red LAN o conectada a Internet mediante el cable de red, desconéctalo de inmediato ya que a mi me sucedieron muchos errores con la instalación del servidor web apache por causa de esto, así que mejor para no tener futuros problemas con la instalación de todos los programas que vamos a utilizar no tengas el cable de red conectado mientras realizas este tutorial. Luego de que se termine la instalación de todos nuestros programas ya podrás conectar nuevamente el cable de red y veras como todo funciona.

ATENCIÓN: esta guía de instalación deja la configuración por defecto de todo el software de servidor mencionado en la misma. No se centra en el afinamiento de la configuración ni detalles sobre seguridad. El objetivo es crear un servidor en el PC para desarrollo y ayuda de los programadores, no para instalar un servidor de cara a su uso en la red. Quien use esta guía de instalación para lo segundo, corre bajo su responsabilidad, nosotros sólo podemos sugerirle que haga los ajustes de seguridad que crea convenientes. En este manual no abordaremos tal tema.

1 - Instalación de Apache - Web oficial: <http://www.apache.org>

Deberemos descargar el Apache en su versión para Windows. Para ello podemos acceder por la siguiente URL: <http://httpd.apache.org/download.cgi> En cualquier caso, podemos encontrar cualquier archivo necesario en la carpeta **httpd/binaries/win32** del servidor que usemos para la descarga (pincharíamos en Other files y eso nos conduciría al servidor seleccionado). Lo que es importante destacar es que lo que hay que descargar son los Binarios (Binary) para Windows (Win32), no los códigos fuente.

Para esta versión, en la fecha de actualización del manual la última versión disponible era la 2.0.55, por lo que el archivo que tenemos que bajar puede ser: **apache_2.0.55-win32-x86-no_ssl.exe** También puede existir el mismo archivo pero con extensión msi, en este caso podríamos bajarlo si disponemos del Windows Installer, la ventaja es que ocupa menos.

Bien, una vez que lo tengamos ejecutamos el instalador y vamos recorriendo las pantallas hasta que salga una donde nos piden unos datos, en cuyo caso pondremos:

Network Domain: 127.0.0.1

Server Name: 127.0.0.1

Administrator's Email Address: nuestro e-mail, aunque no es totalmente necesario.

For All Users, on Port 80, as a Service: seleccionamos esta opción.



La IP 127.0.0.1 es la dirección IP asociada a nuestra máquina, es decir el host local o vulgarmente conocido como **Localhost**. Es importante decir que a la hora de probar tus scripts en modo local, da igual que pongas 127.0.0.1 o localhost. Finalmente, recordar que el Apache se instala por defecto en la carpeta:

C:\Archivos de programa\Apache Group\Apache2

Bien, ahora vamos a instalar el PHP. Asegúrate ahora de tener cerrado el Apache

2 - Instalación de PHP - Web oficial: <http://www.php.net>

2.1 - Copia de archivos

Procedemos a descargar el PHP para Windows. El archivo está localizado en la sección Downloads, apartado Windows Binaries, y es el Zip Package (no el installer, aunque éste ocupe menos luego no nos servirá). Para la versión 5.1.2, que es la versión disponible a fecha de actualización de este manual, es: php-5.1.2-Win32 [8.927KB]. Según vayan saliendo nuevas versiones podrás ir las encontrando en la citada sección downloads.

Una vez descargado todo el ZIP, nos creamos una carpeta en el sitio donde queramos instalar los archivos del servidor (PHP, MySQL...), por ejemplo nos creamos una carpeta en la raíz del disco duro y que quede así: **C:\Servidor** Dentro de esta carpeta nos creamos otra carpeta y la llamamos PHP. Luego extraemos los archivos del ZIP dentro de esa carpeta PHP, tal que los contenidos del ZIP quedarán dentro de la ruta **C:\Servidor\PHP**

Y ahora hay que coger todos los archivos DLL localizados en la carpeta principal **C:\Servidor\PHP** y copiarlos al directorio System (en Windows 9x) o System32 (NT, 2000, XP, 2003) de la carpeta del Windows (los archivos DLL contenidos en la carpeta EXT no hace falta copiarlos).

2.2 - Configuración del archivo php.ini

El siguiente paso es configurar el php.ini. Renombramos o copiamos el archivo **C:\Servidor\PHP\php.ini-dist** y le ponemos **php.ini**. Ahora lo editamos con el mismo block de notas. Si lo necesitáramos (lee antes la explicación), editamos la línea `register_globals = Off` y la colocamos el valor: `register_globals = On`

¿Qué hago con `register_globals`? ¿ON u OFF?

Activar esta directiva nos permite asumir que las variables son globales y pueden llegar por cualquier método (POST, GET, COOKIE, SERVER, etc). Así, por ejemplo, si utilizamos una variable global de sesión o cookie se puede suplantar fácilmente mediante una variable por url, con lo cual nuestro script no es seguro. Un buen programador de PHP tendría la directiva en OFF y usaría los arrays globales (`$HTTP_X_VARS`) o los superglobales `$_POST`, `$_GET`, etc., que están disponibles a partir de la versión 4.1.X de PHP. ¿Y por qué? Pues por esos temas de seguridad en los script y porque debemos acostumbrarnos a no manejar variables globales ya que en un futuro el PHP tendrá la opción en OFF por defecto y no podremos cambiarla. Sin embargo, por temas de compatibilidad con script antiguos o que hagan uso de variables globales, podría interesarnos activar esta característica, pero repito que lo deseable sería tenerla en OFF y hacer uso de los arrays globales o superglobales.

A continuación vamos a indicar a PHP dónde se guardan las extensiones. Dentro del php.ini buscamos **extension_dir** y le ponemos la carpeta que contiene los archivos **php_xxx.dll**, que por defecto es la carpeta ext dentro de PHP. **IMPORTANTE:** durante toda la configuración de directorios, debes utilizar esta barra "/" y no esta "\", además de ponerlo entre comillas. O sea, que debe quedar así:

```
; Directory in which the loadable extensions (modules) reside.  
extension_dir = "C:/Servidor/PHP/ext/"
```

Además podemos activar las extensiones que queramos o necesitemos, para lo cual buscamos **Windows Extensions** y para cargar las extensiones les quitamos el ; de delante. Por ejemplo, si quisiéramos cargar la extensión **gd2.dll** (para manejar las funciones PHP relativas a imágenes) deberíamos cambiar `;extension=php_gd2.dll` por `extension=php_gd2.dll`. Puedes ver para qué sirve cada extensión en el Manual oficial de PHP. Como nota adicional, resaltar que hay algunas extensiones que requieren de librerías extra que no vienen en el paquete completo de PHP, y para hacerlas funcionar tendremos que buscar dichas librerías. No actives todas las librerías a diestro y siniestro porque luego saldrán errores de que no se encuentra tal archivo; antes de instalar una librería, comprueba que en la carpeta de PHP tienes los archivos DLL correspondientes.

IMPORTANTE: la librería **php_mysql.dll** es la que permite manejar las funciones relacionadas con MySQL, y por defecto en PHP5 viene desactivada, por tanto vamos a

activarla de la forma que indicamos antes: buscamos `;extension=php_mysql.dll` y le quitamos el `;` de delante.

Ahora, si vamos a hacer pruebas con upload de archivos vía HTTP, debemos indicar el directorio donde los archivos se almacenarán temporalmente. Para ello buscamos **upload_tmp_dir** y le damos el valor de una carpeta que exista. Por ejemplo, en nuestro directorio `C:\Servidor\PHP\` creamos una carpeta `uploads`, por lo que quedará algo así:

```
; Temporary directory for HTTP uploaded files (will use system default if not
; specified).
upload_tmp_dir = "C:/Servidor/PHP/uploads/"
```

Si queremos cambiar el tamaño máximo de los archivos que pueden subirse vía HTTP, buscamos **upload_max_filesize** y cambiamos el valor por defecto que trae, 2M (2 MB), por el que queramos. No se recomienda poner un valor alto.

```
; Maximum allowed size for uploaded files.
upload_max_filesize = 2M
```

Para trabajar con sesiones, debemos especificar un directorio donde se guarden los archivos temporales. Al igual que 2 pasos antes, buscamos `session.save_path` y le damos el valor de un directorio que exista (o lo creamos):

```
session.save_path = "C:/Servidor/PHP/sessions/"
```

A continuación copiamos el **php.ini** a la carpeta `C:\Windows\` según nuestra versión de Windows.

¡Ahora vamos a decirle al Apache que tenemos el PHP!

2.3 - Configuración del archivo `httpd.conf` de Apache

Hay que editar el archivo **httpd.conf** que encontramos en la carpeta `conf` dentro del directorio del Apache.

Buscamos **Dynamic Shared Object (DSO) Support** que es donde se cargan los módulos. Ahí vamos a cargar el módulo de PHP para Apache, dando la dirección del archivo `php5apache2.dll` que lo contiene. Encontramos hasta un ejemplo:

```
# Example:
# LoadModule foo_module modules/mod_foo.so
#
```

Y ahora añadimos justo debajo:

```
LoadModule php5_module C:/Servidor/PHP/php5apache2.dll
```

De tal manera que tiene que quedar así:

```
# Example:
# LoadModule foo_module modules/mod_foo.so
#
LoadModule php5_module C:/Servidor/PHP/php5apache2.dll
```

El directorio predeterminado para guardar nuestras páginas es el htdocs del Apache:
C:\Archivos de programa\Apache Group\Apache2\htdocs

Pero podemos cambiarlo fácilmente. Buscamos **DocumentRoot** y nos sale esto:

```
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "C:/Archivos de programa/Apache Group/Apache2\htdocs"
```

Nosotros sustituimos la carpeta htdocs del Apache por la que queramos. Todos los archivos que vayamos a probar con el Apache deberán estar localizados en esta carpeta. Por ejemplo nos creamos una carpeta WEB dentro del directorio del servidor:

```
DocumentRoot "C:/Servidor/WEB/"
```

Por lo que construiremos toda nuestra página dentro de ese directorio, que equivale al directorio raíz de nuestro host local.

Ahora buscamos el **DirectoryIndex** y nos sale algo como esto:

```
#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
# The index.html.var file (a type-map) is used to deliver content-
# negotiated documents. The MultiViews Option can be used for the
# same purpose, but it is much slower.
#
DirectoryIndex index.html index.html.var
```

Modificamos la última línea para que si entramos en un directorio tipo **http://127.0.0.1/directorio/** nos auto ejecute el index predeterminado (si existe) y no nos salga un mensaje de error. Puedes usar más nombres si lo prefieres. Se ejecutarán por orden: si no encuentra el primero, pasa al segundo; si no está el segundo, pasa al tercero, y así sucesivamente hasta que si no encuentra ninguno entonces da error. En la siguiente línea, si en nuestro directorio tenemos un index.htm y un index.php, por defecto se ejecutará el index.htm ya que está antes.

```
DirectoryIndex index.html index.htm index.php index.php3 index.php4 index.php5
index.phtml index.html.var
```

Ahora le añadimos debajo estas líneas:

```
AddType application/x-httpd-php .php .php3 .php4 .php5 .phtml
AddType application/x-httpd-php-source .phps
```

La primera indica las extensiones que serán interpretadas por el Apache. Por ejemplo podemos añadir la extensión .htm o .html para que el Apache ejecute el código PHP contenido en esas páginas (es decir, el uso de código PHP no está limitado exclusivamente a archivos *.PHP). La segunda sirve para que si entras en una página **loquesea.php** entonces se muestra el código PHP a color, muy útil si queremos por ejemplo mostrar el código fuente a color en una página. De tal forma que al final todo queda así:

```
#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
# The index.html.var file (a type-map) is used to deliver content-
# negotiated documents. The MultiViews Option can be used for the
# same purpose, but it is much slower.
#
DirectoryIndex index.html index.htm index.php index.php3 index.php4 index.php5
index.phtml index.html.var
AddType application/x-httpd-php .php .php3 .php4 .php5 .phtml
AddType application/x-httpd-php-source .phps
```

Y ya podemos guardar el archivo httpd.conf

2.4 - Probando nuestro servidor Apache


Ahora vamos a probar el Apache y PHP, para lo cual deberemos arrancar el Apache.


Para Apache 2.0.X accederemos desde los accesos directos que se crean en el menú de inicio al acceso directo **Monitor Apache Servers** y aparecerá un icono al lado del reloj. Pinchando en el icono, sale un menú desde donde podremos iniciar, apagar y reiniciar el Apache. Como queremos encenderlo, si no lo hace automáticamente le daremos a Start. Si ya estaba encendido le daremos a Restart (útil si cambiamos alguna configuración de PHP "al vuelo") para que al reiniciarlo nos coja los cambios que hemos hecho.

Si todo ha ido bien, podemos crear un archivo llamado por ejemplo **info.php** dentro de la carpeta de nuestra web cuyo contenido sea exclusivamente el siguiente:

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

Si accedemos mediante nuestro navegador a la dirección **http://127.0.0.1/info.php** o **http://localhost/info.php** (recuerda que 127.0.0.1 = localhost) con el Apache activo deberemos visualizar una página de información de PHP. Si no sale, algo ha fallado, revisa los pasos.

PHP Version 5.0.3	
	
System	Windows NT FAMILIAR 5.1 build 2600
Build Date	Dec 15 2004 08:06:41
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--with-gd=shared"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS\php.ini
PHP API	20031224
PHP Extension	20041030
Zend Extension	220040412
Debug Build	no
Thread Safety	enabled
IPv6 Support	enabled
Registered PHP Streams	php, file, http, ftp, compress.zlib
Registered Stream Socket Transports	tcp, udp

<p>This program makes use of the Zend Scripting Language Engine: Zend Engine v2.0.3, Copyright (c) 1998-2004 Zend Technologies</p>	<p>Powered By</p> 
---	--

Como has podido observar, para acceder a nuestro servidor local y así poder probar nuestros script y las páginas de nuestra web, siempre accederemos con la dirección **127.0.0.1 o localhost** (sin WWW) en el navegador pues esta es la dirección del host local de nuestro ordenador. En general, para visualizar o acceder a cualquier archivo (PHP, HTML, TXT, ZIP, GIF, JPG, etc.) que tengamos en nuestra carpeta de la web, lo haremos de la siguiente forma:

<http://localhost/CARPETA/ARCHIVO.EXT>

Donde CARPETA es la carpeta (o serie de carpetas) relativas a la raíz en que se encuentra el ARCHIVO.EXT

Por tanto resaltar que no sólo podemos hacer llamadas a script PHP, sino también a páginas HTML, descarga de archivos, imágenes, etc., tan solo poniendo la dirección correcta en el navegador. Es simplemente un servidor. De esta forma, podemos tener nuestra web funcionando en nuestro propio ordenador para construirla desde cero o bien hacer las pruebas que necesitemos sin tener que estar conectados a Internet.

3 - Instalación de MySQL - Web oficial: <http://www.mysql.com>

Vamos a hacer la instalación de una versión 5.0.X, por lo que en este caso buscaremos la versión que corresponda en los downloads para Windows. En la fecha de actualización de este manual, la versión utilizada fue la 5.0.19 Y repito que usamos una versión 5.0.X, no una 4.1.X, pues la 4.1.X no funcionará si sigues los pasos de este tutorial.

Existen dos tipos de archivo descargable, en ZIP y en EXE. Yo prefiero el ZIP porque es descomprimir y listo, y luego para actualizar la versión es más sencillo. El instalador EXE lleva un setup muy majo, por defecto se instala en **C:\MySQL** Recomiendo elegir otra ruta de instalación, y por ejemplo lo vamos a hacer en **C:\Servidor\MySQL** para tener el PHP y MySQL en la misma carpeta.

Por defecto, MySQL crea un usuario de nombre root y con contraseña, y éste es el usuario que usaremos en nuestros script o en las aplicaciones que nos lo pidan (foros, PHP Nuke, etc.).

4 - Instalación de phpMyAdmin - Web oficial: <http://www.phpmyadmin.net>

PhpMyAdmin es una utilidad que nos sirve para interactuar con una base de datos de forma muy sencilla y desde una interfaz web. Nos sirve por ejemplo para crear bases de datos, tablas, borrar o modificar datos, añadir registros, hacer copias de seguridad, etc. Es una aplicación tan útil que casi todos los hosting con MySQL disponen de ella, por ello se analizará su instalación. Al ser una aplicación escrita en PHP, necesita de Apache y MySQL para poder funcionar.

Si accedemos a la página oficial, nos encontramos en la sección de downloads con una lista de links, y el que tenemos que elegir es el ZIP de la última versión. En nuestro caso vamos a utilizar la versión 2.8.0.3 de esta aplicación, que es la versión disponible en la fecha de actualización de este manual.

La instalación es relativamente sencilla: extraer todo el ZIP en la carpeta raíz de nuestra web. OJO: en la carpeta de nuestra web, **C:\Servidor\WEB** y no dentro de la carpeta del PHP o similares. Como al terminar la extracción de archivos la carpeta donde están contenidos tiene un nombre extraño, mejor la renombramos a **phpmyadmin**, de forma que quedaría algo así: **C:\Servidor\WEB\phpmyadmin**

Ahora vamos a hacer una pequeña configuración del phpMyAdmin. Para ello debemos abrir el archivo **config.default.php** el cual se encuentra ubicado en **C:\Servidor\WEB\phpmyadmin\libraries** y guardarlo como **config.inc.php** en **C:\Servidor\WEB\phpmyadmin** y, leyendo de arriba hacia abajo, buscamos la primera aparición de la siguiente línea:

```
$cfg['PmaAbsoluteUri'] = '';
```

Debemos darle la ruta absoluta donde tenemos el phpMyAdmin. En nuestro caso sería así:

```
$cfg['PmaAbsoluteUri'] = 'http://localhost/phpmyadmin/';
```

Recuerda que podemos poner localhost o 127.0.0.1 (a gusto del consumidor). Ahora buscamos si nos aparece algo como esto:

```
$cfg['blowfish_secret'] = '';
```

Le pondremos una cadena de caracteres cualquiera, que servirá de semilla para la encriptación de contraseñas al usar la autenticación con cookies:

```
$cfg['blowfish_secret'] = 'aquí puedes poner lo que quieras';
```

Mas abajo encontraras:

```
$cfg['Servers'][$i]['password'] = '';
```

En esta variable debes colocar tu contraseña de root, la cual definiste previamente en la instalación de MySQL, porque de lo contrario phpMyAdmin te arrojara un error al momento de probarlo.

```
$cfg['Servers'][$i]['password'] = 'tu contraseña';
```

Y guardaremos el archivo, pero no lo cerraremos. Puedes probar phpMyAdmin para ver si funciona, ya sabes: <http://localhost/phpmyadmin/>

6 - Cómo realizar las actualizaciones a versiones superiores

Como nos gusta ir a la última y continuamente hay actualizaciones de seguridad, voy a poner unas notas de cómo actualizar estos elementos una vez los tengamos instalados y saquen una nueva versión. Ten en cuenta que al sacar una nueva versión puede que haya cambios grandes en la forma de instalación, pues estas notas se harán suponiendo que la forma de instalación no cambia de una versión a otra, lo cual es habitual.

6.1 - Cómo actualizar Apache

Para actualizarlo lo primero será cerrar todos los procesos que usan Apache, incluidos los que se ejecutan ocultos como servicio del sistema. Lo más sencillo es ir al menú CTRL + ALT + Supr y cerrar todos los procesos en que esté involucrado el Apache. Después desinstalaremos el programa desde el Panel de Control - Agregar o Quitar Programas. A continuación borrarémos manualmente aquellas carpetas del Apache que el desinstalador no haya borrado. Y finalmente, instalaremos la nueva versión de Apache y configuraremos el httpd.conf igual que se explica en los pasos anteriores, usando las mismas carpetas que usaste en la versión anterior de Apache para que nada cambie.

6.2 - Cómo actualizar PHP

Empezaremos cerrando todos los procesos de Apache y MySQL de la forma citada en el párrafo anterior. A continuación eliminaremos la carpeta PHP que se encuentra en C:\Servidor\PHP\ Luego instalaremos la nueva versión de PHP como se dice arriba, configuraremos el nuevo php.ini, y en los pasos que se dice que copiemos ciertos archivos a los directorios de Windows, sobrescribiremos todos archivos existentes con los nuevos, incluido el nuevo php.ini Ya sólo te queda arrancar de nuevo el Apache y MySQL y ver si todo ha ido bien.

6.3 - Cómo actualizar MySQL

Al igual que antes, cerraremos todos los procesos activos que tengan que ver con MySQL para evitarnos problemas. Después copiaremos el directorio **data** (que está

dentro de la carpeta de MySQL y es el que contiene nuestras bases de datos, tablas, etc. que queremos guardar) a otro directorio cualquiera de forma temporal. Luego desinstalaremos MySQL, ya sea desde el Agregar o Quitar Programas si usaste la versión que trae instalador, o bien borrando el directorio si usaste la versión que viene sin instalador. A continuación instalaremos la nueva versión, tras lo cual volveremos a copiar el directorio **data** de nuevo a la carpeta de MySQL sobrescribiendo el que haya, y de esta forma no se pierden los datos de nuestras bases de datos. Y finalmente sólo queda reiniciar MySQL y comprobar si los script y phpMyAdmin siguen funcionando.

6.4 - Cómo actualizar phpMyAdmin

Éste es el más sencillo de actualizar, pues lo único que hay que hacer es borrar el anterior y luego instalar y configurar el nuevo como se cita más arriba. Además, asegúrate también de cambiar el usuario y contraseña para que luego funcione.

Tomado del web site <http://www.maestrosdelweb.com/editorial/phpmysqlap/> y con algunas modificaciones hechas por mi.

Cualquier inquietud o alguna sugerencia para mejorar este tutorial por favor escribir a: juanpablo.ortegonreyes@gmail.com

Juan Pablo Ortegón Reyes
28 de abril de 2006