Programación Web con PHP y MySQL– Nivel 2

Unidad 4: Introducción a PHP

Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 4: Introducción a PHP - Variables

Objetivos

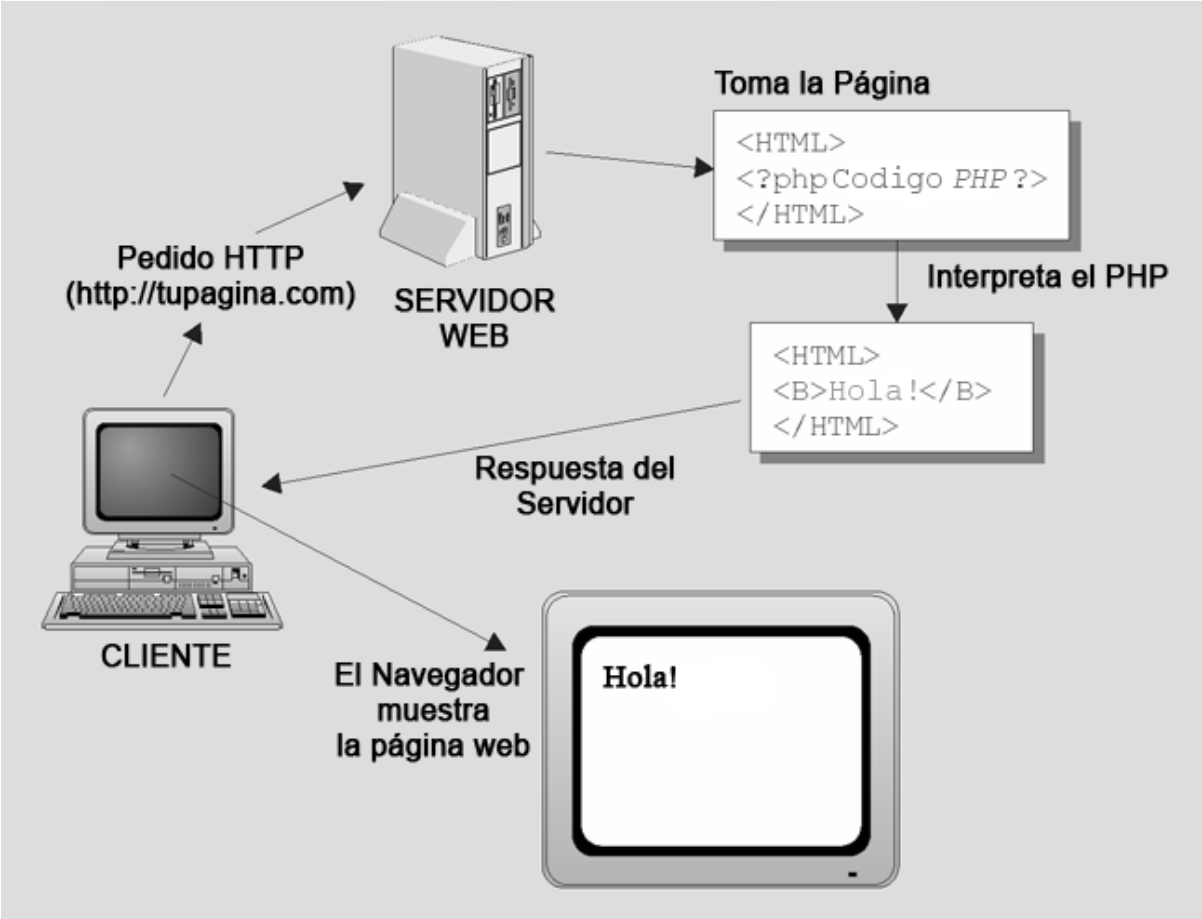
Que el alumno logre:

- Aprender los conceptos de tabla, columna, fila o registro y valor de un dato dentro de la tabla.
- Aprender a definir una llave o clave primaria y cuál es el propósito de la misma.

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.

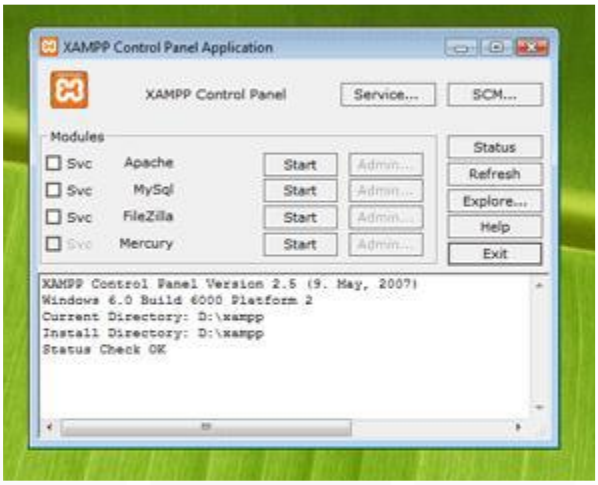
Si se van a usar bases de datos (generalmente **MySQL**) también hay que asegurarse de que el hosting lo soporte.

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.

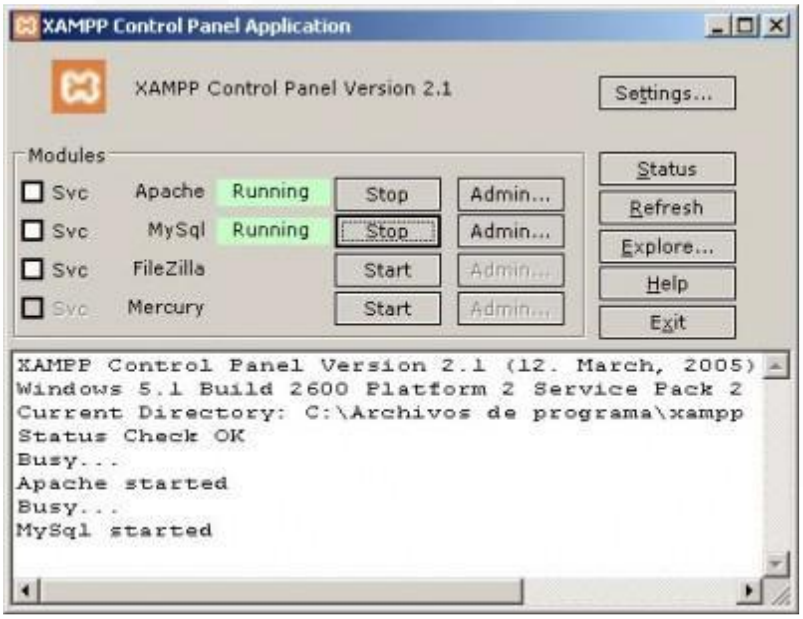
Puede descargarse de http://www.apachefriends.org/es/xampp.html

(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:



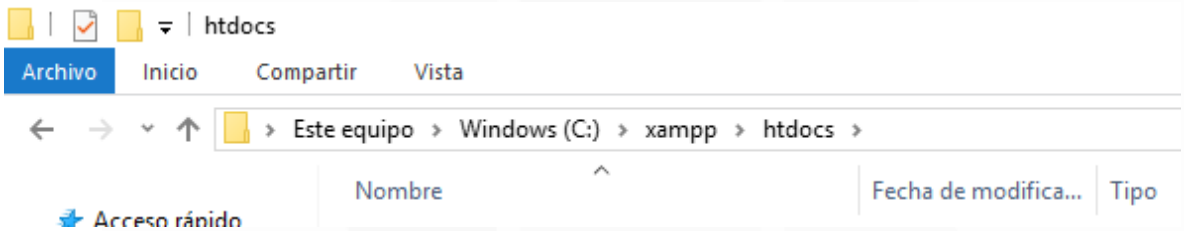
Lo normal es activar los dos primeros, para ello hacemos click en los botones que dicen “Start” pertenecientes a Apache y MySQL, debería aparecer la palabra “Running” al costado de cada uno de ellos, como podemos ver a continuación:



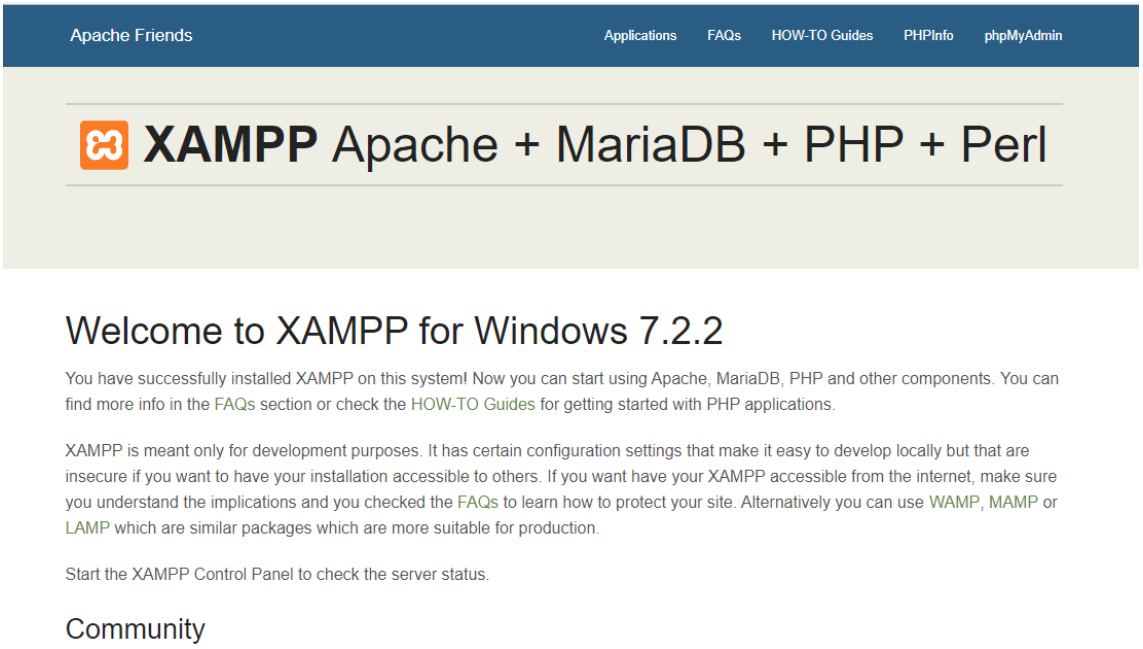
Habilitar estas dos opciones nos permitirá que desde cualquier navegador tecleemos la IP local (http://127.0.0.1, o bien http://localhost) y accedamos a la verdadera interfaz de la aplicación, desde la cual tendremos acceso a nuestras bases de datos MySQL a través del phpMyAdmin.

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.



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/dashboard/phpinfo.php y le damos enter

Deberíamos obtener como respuesta una página como la siguiente:

PHP Version 7.2.2

System	Windows NT LAPTOP-EO51LA05 10.0 build 17134 (Windows 10) i586
Build Date	Jan 31 2018 19:27:55
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x86\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x86\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\WINDOWS
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,TS,VC15
PHP Extension Build	API20170718,TS,VC15
Debug Build	no
Thread Safety	enabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring

¿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>
<html>
<head>
<meta charset="utf-8" />
<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>
<html>
<head>
<meta charset="utf-8" />
<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>
<html>
<head>
<meta charset="utf-8" />
<title>Hola mundo</title>
</head>
<body>
<?php
echo "Este es
el
clásico "; echo "Hola mundo";
?>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" />
<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>
<html>
<head>
<meta charset="utf-8" />
<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();
?>
```

Resumen

En esta Unidad…

En la presente unidad trabajamos con los conceptos básicos del lenguaje PHP

En la próxima Unidad…

En la próxima unidad trabajaremos con variables y operadores.