Desarrollo de Aplicaciones Web I

Tema 7 - Lenguaje PHP

Índice

1. Introducción al lenguaje PHP

PHP, acrónimo de "PHP: Hypertext Preprocessor", es un lenguaje "Open Source" interpretado, de alto nivel, especialmente pensado para desarrollos Web, embebido en páginas HTML y ejecutado en el Servidor. La mayoría de su sintaxis es similar a C, Java y Perl y es fácil de aprender. La meta de este lenguaje es permitir escribir a los creadores de páginas Web, páginas dinámicas de una manera rápida y fácil, aunque se pueda hacer mucho más con PHP.

Entre sus principales características cabe destacar su potencia, su alto rendimiento, su facilidad de aprendizaje, su escasez de consumo de recursos, su gran número de librerías de funciones, y su amplio uso en Internet para la generación de portales dinámicos.

Toda la documentación necesaria para trabajar con PHP se puede encontrar en la Web oficial del lenguaje:
http://www.php.net

Con PHP puede hacer cualquier cosa que se pueda hacer con un script CGI, como procesar la información de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies. Y esto no es todo, se puede hacer mucho más.

Existen tres campos en los que se usan scripts escritos en PHP:

- **Scripts del lado del servidor**. Este es el campo más tradicional y el principal foco de trabajo. Se necesitan tres cosas para que esto funcione. El intérprete PHP (CGI ó módulo), un servidor Web y un navegador. Es necesario lanzar la ejecución del servidor Web con PHP instalado. El resultado del programa PHP se puede obtener a través del navegador, conectándose con el servidor Web.

- **Scripts en la línea de comandos**. Pueden crearse scripts PHP y ejecutarlos sin ningún servidor Web o navegador. Solamente necesita el intérprete PHP para usarlo de esta manera. Este tipo de uso es ideal

para scripts ejecutados regularmente desde “cron” (en sistemas Unix o Linux) o el “Planificador de tareas” (en Windows). Estos scripts también pueden ser usados para tareas simples de procesamiento de texto gracias a la cantidad de funciones disponibles en el lenguaje.

- **Escribir aplicaciones de interfaz gráfica**. Probablemente PHP no sea el lenguaje más apropiado para escribir aplicaciones gráficas, pero si conoce bien PHP, y quisiera utilizar algunas características avanzadas en programas clientes, puede utilizar PHP-GTK para escribir dichos programas. También es posible escribir aplicaciones independientes de la plataforma. PHP-GTK es una extensión de PHP, no disponible en la distribución principal. Si está interesado en PHP-GTK, puedes visitar las páginas Web del proyecto en http://gtk.php.net.

Utilización de PHP:

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, como 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, como Apache, Microsoft Internet Information Server, Personal Web Server, Netscape e iPlanet, Oreilly Website Pro server, Xitami, OmniHTTPd y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, y también, para aquellos otros que soporten el estándar CGI, PHP puede usarse como procesador CGI.

De modo que, con PHP se 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. Muchas bibliotecas y aplicaciones grandes (incluyendo la biblioteca PEAR) están escritas íntegramente usando programación orientada a objetos.

Características de PHP:

PHP no se limita solo 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 (en tiempo de ejecución). 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 son un ejemplo de las que están soportadas:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	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 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 si fueran 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, se soportan los estándares SAX y DOM. Puede utilizar la extensión XSLT para transformar documentos XML.

Si se usa PHP en el campo del comercio electrónico, son muy útiles las funciones Cybercash, CyberMUT, VeriSign Payflow Pro y CCVS para la creación de 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), tratamiento de imágenes, conversión de calendarios, traducción de textos, ...

Como se puede apreciar, este resumen no es suficiente para enumerar todas las características y beneficios que PHP ofrece, con lo que se debe consultar la referencia de los módulos y funciones disponibles en PHP en el extenso manual de la Web oficial:

http://www.php.net/manual/es/funcref.php

El modelo de ejecución de PHP:

El funcionamiento de las páginas en PHP alojadas en un servidor es muy sencillo...

- El navegador del cliente solicita el documento PHP.

- Llega la solicitud del servidor y el servidor localiza el documento (el script PHP), lanza el intérprete del lenguaje y ejecuta todo su código.

- Durante la ejecución del código se genera el resultado en HTML, que puede ser almacenado en un buffer interno el cual, cuando se llena o se termina el script, se devuelve al servidor para que lo transfiera al cliente.

- El servidor transfiere el resultado en HTML y es mostrado en el navegador del cliente.

Cómo crear Páginas PHP, un Ejemplo:

Como ocurre con otros lenguajes de script, los archivos ".php" son archivos de texto normales, por lo que no es necesario ningún editor especial para crearlos, puede usarse cualquier editor que genere código ASCII y a ser posible también con codificación UTF-8.

Un archivo ".php" puede contener texto, código HTML, código PHP o cualquier combinación de estos. Si no contiene código PHP se comporta como un archivo ".html" normal.

Se debe tener en cuenta que todos los archivos ".php" requieren una parte de proceso por el servidor (para cargarlo e interpretarlo), por lo cual no es conveniente renombrar a ".php" los archivos que no contengan código. Esta recomendación se puede pasar por alto teniendo instaladas en

el servidor librerías dedicadas a la aceleración de carga, compilación y optimización de código de los sripts PHP, como puede ser “Zend Engine” o similares.

```
<?php
  session_start();
  $numero= 0;
  if (isset( $_SESSION['numero'])) {
    $numero= (int)$_SESSION['numero'];
    $numero++;
  }
  $_SESSION['numero']= $numero;
?>
<html>
  <head><title>Hola y números. Ejemplo PHP</title></head>
<body bgcolor="white">
Hola, mundo. Te lo repito (<?php echo $numero; ?>)
<?php if ($numero == 1) { ?>
 vez
<?php } else { ?>
 veces
<?php }//if ?>
</body></html>
```

Los archivos PHP se pueden ubicar en cualquier lugar del sistema de ficheros que sea accesible para el servidor Web, ahora bien, para que sean accesibles desde un navegador Web, deben estar localizadas en alguna carpeta que sea pública desde el servidor Web, es decir, visible para internet.

Ubicando los archivos PHP en una carpeta pública del servidor Web, e invocando la URL correspondiente desde un navegador Web es suficiente para tener acceso a una aplicación Web.

La carpeta pública del servidor Web puede ser un directorio específico para una sola aplicación Web, con lo que dentro de éste, se colocarán todos los archivos “.php” y recursos necesarios (organizados de la forma que deseemos) para que se pueda invocar desde un navegador cliente.

Generalmente los archivos iniciales dentro del directorio de la aplicación Web suelen ser los siguientes, dados en orden de búsqueda: “index.php”, “index.html”, “index.htm”, pero esto se puede modificar en la configuración del servidor Web.

2. Integración de PHP y HTML.

El lenguaje PHP se integra dentro del código HTML, un archivo o script PHP de hecho, **siempre** es interpretado como archivo HTML hasta que se encuentran unas etiquetas especiales de comienzo y final.

Etiqueta de comienzo de modo PHP	→	<?php
Etiqueta de fin de modo PHP	→	?>

Estas etiquetas delimitan el código PHP dentro del archivo de texto que es interpretado inicialmente como texto de salida HTML, con lo que para evitar resultados inesperados, se deben evitar espacios en blanco y saltos de línea al principio del fichero si lo que se desea es generar desde PHP otra salida que no sea HTML.

La etiqueta de fin de modo PHP se puede omitir si es lo último que vamos a tener en el fichero, así también nos aseguramos espacios y saltos de línea que pueden afectar a la salida HTML.

Existen otras formas de indicar las etiquetas de comienzo y fin de modo PHP, pero no suelen ser utilizadas porque, o son muy largas de escribir, o hay que configurar algunas propiedades de PHP en el servidor Web, lo cual no suele estar accesible a todos los desarrolladores.

- Etiqueta abreviada de comienzo de modo PHP "<?" la cual requiere configuración en el archivo PHP.INI del servidor Web para que sea interpretada adecuadamente (variable “short_open_tag” a “On”).

- Etiquetas HTML <script languaje="php"> </script>, que están enfocadas a servidores Web de Microsoft, y no son aceptadas en otros tipos de servidores, y además, suelen olvidarse fácilmente por ser largas de escribir.

Un Ejemplo Sencillo:

```
<html>
    <head>
        <title>Ejemplo Sencillo</title>
    </head>
    <body>
        <?php
        echo "Hola, &iexcl;soy un script PHP!";
        ?>
    </body>
</html>
```

Puede apreciarse que un script PHP no es lo mismo que un script escrito en otro lenguaje de programación como Perl o C. En vez de escribir un programa con muchos comandos para crear una salida en HTML, escribimos el código HTML con cierto código PHP embebido (incluido) en el mismo, que producirá cierta salida (en el ejemplo anterior, producirá un texto).

Lo que distingue a PHP de la tecnología Javascript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si tuviésemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente solamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar qué código ha producido el resultado recibido. El servidor web puede ser incluso configurado para que procese todos los archivos HTML con PHP.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales.

¡¡ No sintáis miedo de PHP, en poco tiempo podréis empezar a escribir vuestros primeros scripts !!

Algo útil - Script de Información:

Veamos ahora algo que puede sernos útil. Vamos a ver la información del sistema PHP y del Servidor Web que lo aloja e interpreta.

Crearemos un archivo “info.php” en el que escribiremos:

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

Este archivo se debe guardar en el directorio público del servidor Web, para que posteriormente, al acceder desde el navegador a la dirección de su máquina local, y hacer la petición HTTP, ésta sea visible. Para abrir el archivo creado en el navegador, tendríamos que acceder a http://localhost/info.php, o a la dirección UTRL donde tengamos almacenado el script.

Una vez abierta la URL, nos mostrará la información del servidor, del módulo PHP, del entorno del Sistema Operativo, de las características del Navegador Web, y de otros datos que el Navegador envía al Servidor Web en las peticiones HTTP. Toda esta información es guardada en variables y arrays de PHP.

3.Programar en PHP

Sintaxis básica:

Como ya se ha comentado anteriormente, un Script PHP es un archivo de texto, que nada mas abrirse se interpreta como texto HTML, y para entrar en modo PHP hay que utilizar las etiquetas de comienzo "**<?php**" y fin "**?>**" de PHP.

Separación de instrucciones:

Las separación de instrucciones se hace de la misma manera que en C o Perl, terminando cada declaración con el carácter punto y coma "**;**".

Comentarios:

PHP soporta el estilo de comentarios de 'C', 'C++' y de la interfaz de comandos de Unix.

```
//Esto es un comentario de Línea.
/* Esto es un comentario de bloque
   que acaba cuando se cierra.
*/
# Esto es un comentario de Unix.
```

Los estilos de comentarios de una línea actualmente sólo comentan hasta el final de la línea o del bloque actual de código PHP, lo primero que ocurra. Esto implica que el código HTML tras poner "// ?>" será impreso ya que "?>" sale del modo PHP, retornando al modo HTML, el comentario "//" no le influye.

Tipos de Datos:

PHP soporta ocho tipos primitivos.

Cuatro tipos escalares:

- boolean
- integer

- float (número de punto-flotante, también conocido como 'double')
- string

Dos tipos compuestos:
- array
- object

Y finalmente dos tipos especiales:
- resource
- NULL

El tipo de una variable usualmente no es declarado por el programador; en cambio, es decidido en tiempo de compilación por PHP dependiendo del contexto en el que es usada la variable.

Las variables en PHP siempre comienzan con un signo de dólar ("$").

Nota: Si se desea chequear el tipo y valor de una cierta expresión, usar la función "var_dump()".
Si tan solo se desea una representación legible para humanos del tipo de dato, para propósitos de depuración, usar "gettype()". Para chequear por un cierto tipo, no utilizar "gettype()"; en su lugar utilizar las funciones "is_<tipo>()". Algunos ejemplos:

```
<?php
$bool = TRUE;   // un valor booleano
$str  = "foo";  // una cadena
$int  = 12;     // un entero

echo gettype($bool); // imprime "boolean"
echo gettype($str);  // imprime "string"

// Si este valor es un entero, incrementarlo en cuatro
if (is_int($int)) { $int += 4; }

// Si $bool es una cadena, imprimirla
// (no imprime nada)
if (is_string($bool)) {
    echo "Cadena: $bool";
}
?>
```

Moldeamiento de Tipos:

PHP no requiere (o soporta) la definición explícita de tipos en la declaración de variables; el tipo de una variable es determinado por el contexto en el que la variable es usada. Lo que quiere decir que si asigna un valor de cadena a la variable "$var", entonces "$var" se convierte en una cadena. Si luego asigna un valor entero a "$var", ésta se convierte en entera.

Si se desea forzar que una variable sea evaluada como un cierto tipo, se debe Moldear su tipo.

El moldeamiento de tipos en PHP funciona de forma muy similar a como ocurre en C: el nombre del tipo deseado es escrito entre paréntesis antes de la variable que debe ser moldeada.

Los moldeamientos permitidos son:

- (int), (integer) - moldeamiento a entero
- (bool), (boolean) - moldeamiento a booleano
- (float), (double), (real) - moldeamiento a flotante
- (string) - moldeamiento a cadena
- (array) - moldeamiento a matriz
- (object) - moldeamiento a objeto

Variables:

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas. Los nombres de variables siguen las mismas reglas que otras etiquetas en PHP. Un nombre de variable valido tiene que empezar con una letra o un carácter de subrayado (guión bajo o underscore), seguido de cualquier número de letras, números y subrayados.

Variables predefinidas:

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. De todas formas, muchas de esas variables no pueden estar completamente documentadas ya que dependen de sobre qué servidor se esté ejecutando, la versión y configuración de dicho servidor,

y otros factores. Algunas de estas variables no estarán disponibles cuando se ejecute PHP desde la línea de comandos.

PHP superglobales:

Este tipo de variables están definidas siempre y son accesibles desde cualquier script que se ejecute. Son variables de tipo “array”, y se accede a sus valores por medio del nombre clave de las variables que almacena.

- **$GLOBALS**: Contiene una referencia a cada variable disponible en el espectro de las variables del script. Las claves de esta matriz son los nombres de las variables globales.

- **$_SERVER**: Variables definidas por el servidor Web ó directamente relacionadas con el entorno en donde el script se esta ejecutando.

- **$_GET**: Variables proporcionadas al script por medio de la URL y/o en una petición HTTP GET.

- **$_POST**: Variables proporcionadas al script por medio de una petición HTTP POST.

- **$_COOKIE**: Variables proporcionadas al script por medio de HTTP COOKIES.

- **$_FILES**: Variables proporcionadas al script por medio de la subida de ficheros vía HTTP.

- **$_ENV**: Variables proporcionadas al script por medio del entorno de ejecución que brinda el Sistema Operativo donde está el Servidor Web.

- **$_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 es definido por la directiva de configuración “variables_order”.

  Nota: Cuando se utiliza la línea de comandos, “argv” y “argc” no son incluidas aquí; estas variables se podrán encontrar en la matriz de sesión.

- **$_SESSION**: Variables registradas en la sesión del usuario en cualquier momento y en cualquier script. Existen mientras no finalice el tiempo de vida de la sesión (caduque) y son accesibles desde cualquier script.

Constantes:

Se puede definir una constante usando la función "define()". Una vez definida, no puede ser modificada ni eliminada. Solo se puede definir como constantes valores escalares (boolean, integer, float y string).

Para obtener el valor de una constante solo es necesario especificar su nombre. A diferencia de las variables, no se tiene que especificar el prefijo "$". También se puede utilizar la función "constant()", para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica usa la función "get_defined_constants()" parar obtener una lista de todas las constantes definidas.

Nota: Las constantes y las variables (globales) se encuentran en un espacio de nombres distinto. Esto implica que por ejemplo TRUE y $TRUE son diferentes.

Si se usa una constante todavía no definida, PHP asume que se está refiriendo al nombre de la constante en si, y se lanzará un aviso si esto sucede. Se puede usar la función "defined()" para comprobar la existencia de dicha constante.

Constantes Predefinidas:

PHP ofrece un largo número de constantes predefinidas a cualquier script en ejecución. Muchas de estas constantes, sin embargo, son creadas por diferentes extensiones, y solo estarán presentes si dichas extensiones están disponibles, bien por carga dinámica o porque han sido compiladas. Se puede encontrar una lista de constantes predefinidas en la dirección...

http://www.php.net/manual/es/reserved.constants.php

Expresiones:

Las expresiones son la piedra angular de PHP. En PHP, casi cualquier cosa que se escribe es una expresión. La forma más simple y ajustada de definir una expresión es la frase *"cualquier cosa que tiene un valor"*.

Las formas más básicas de expresiones son las constantes y las variables. Cuando se escribe "$a = 5", estás asignando el valor "5" a la variable "$a". "5", obviamente, tiene el valor 5 ó, en otras palabras "5" es una expresión con el valor 5 (en este caso, "5" es una constante entera).

Después de esta asignación, se espera que el valor de "$a" sea 5 también, de manera que si se escribe "$b = $a", se espera también que se comporte igual que si se escribiese "$b = 5". En otras palabras, "$a" es una expresión también con el valor 5. Si todo va bien, eso es exactamente lo que pasará.

Operadores:

Un operador es algo a lo que se le entrega uno o más valores (o expresiones, en jerga de programación) y produce otro valor (de modo que la construcción misma se convierte en una expresión).

Así que puede pensar sobre las funciones o construcciones que devuelven un valor (como "print()") como operadores, y en aquellas que no devuelven nada (como "echo") como cualquier otra cosa.

Existen tres tipos de operadores:

- En primer lugar se encuentra el operador unario, el cual opera sobre un único valor, por ejemplo **!** (el operador de negación) o **++** (el operador de incremento).
- El segundo grupo se conoce como operadores binarios; este grupo contiene la mayoría de operadores que soporta PHP, y una lista se encuentra disponible más adelante en la sección "Precedencia de Operadores".
- El tercer grupo consiste del operador ternario: (condición **?** expresión **:** expresión). Éste debe ser usado para evaluar una condición y según sea verdadera o falsa, seleccionar una de las dos expresiones que aparecen a continuación, siendo la primera expresión la que se devuelve si la condición resulta verdadera, y devolviéndose la segunda expresión si la condición resulta falsa. Rodear las expresiones ternarias con paréntesis

es una muy buena idea para mejorar su legibilidad, aunque no es necesario para el intérprete de PHP.

Asociatividad	Operadores	Información Adicional
no-asociativo	new	Creación de objetos
izquierda	[	Acceso a array()
no-asociativos	++ --	Incremento/decremento
no-asociativos	! ~ - (int) (float) (string) (array) (object) @	Negación lógica y binaria, cambio de signo, conversión de tipos, desactivar alertas PHP
izquierda	* / %	Aritmética
izquierda	+ - .	Adición aritmética, y cadena
izquierda	<< >>	manejo de bits
no-asociativos	< <= > >=	Comparación lógica
no-asociativos	== != === !==	Comparación lógica
izquierda	&	manejo de bits, y referencias
izquierda	^	manejo de bits
izquierda	|	manejo de bits
izquierda	&&	Y lógico
izquierda	||	O lógico
izquierda	? :	Operador ternario
derecha	= += -= *= /= .= %= &= |= ^= <<= >>=	Asignación directa y asignación con operaciones aritméticas y de bits
izquierda	and	Y lógico
izquierda	xor	O exclusivo lógico
izquierda	or	O lógico
izquierda	,	varios usos, separador de datos en argumentos, arrays, ...

La asociatividad de izquierda quiere decir que la expresión es evaluada desde la izquierda a la derecha, la asociatividad de derecha quiere decir lo contrario.

Estructuras de Control:

Todo script PHP se compone de una serie de sentencias. Una sentencia puede ser una asignación, una llamada a función, un bucle, una sentencia condicional e incluso una sentencia que no haga nada (una sentencia vacía).

Las sentencias normalmente acaban con punto y coma.

Además, las sentencias se pueden agrupar en grupos de sentencias encapsulando un grupo de sentencias entre llaves "**{**" y "**}**". Un grupo de sentencias es también una sentencia.

Las Estructuras de Control de PHP son similares a las estructuras de “C”.

```
if (condición) sentencia;

else sentencia;

elseif (condición) sentencia;

while (condición) sentencia;

do { sentencia; } while (condición);

for (iniciaciones; condición-mientras; incrementos)
sentencia;

foreach (expresion_array as $value) sentencia;
foreach (expresion_array as $key => $value) sentencia;
```

`break` → escapa de la estructuras de control iterante (bucle) actuales *for*, *while*, o *switch*.

`continue` → se usa dentro de la estructura del bucle para saltar el resto de la iteración actual del bucle y continuar la ejecución al comienzo de la siguiente iteración.

`switch (expresión) / case opción: / default:`

`return (argumento);` → Termina inmediatamente la ejecución de la función y retorna su argumento como valor de la función. "return()" también terminará la ejecución de una sentencia "eval()" o de un script PHP incluido.

`require() / include()` → La sentencia "require()" o "include()" incluyen y evalúan el archivo especificado. Son idénticas en todos los aspectos excepto en el modo de actuar ante un error. "include()" produce un "Warning" continuando la ejecución, mientras que "require()" produce un "Error Fatal", terminando la ejecución.

`require_once() / include_once()` → Estas funciones incluyen y evalúan el fichero especificado durante la ejecución del script. Se comporta de manera similar a las anteriores, con la única diferencia que si el código ha sido ya incluido, no se volverá a incluir. Esto es útil para incluir scripts en los que sólo se definen funciones o clases dentro de ellos, sin tener código ejecutable directamente.

Funciones:

Son piezas de código que realizan una acción perfectamente definida por medio de unos datos enviados como argumentos de función, y devuelve un valor con el resultado del proceso.

Una función se puede definir con la siguiente sintaxis:

```
<?php
function ejemplo( $arg_1, $arg_2, ..., $arg_n)
{
    echo "Funcion de ejemplo.\n";
    ...instrucciones...
    return $retval;
}
?>
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo de la función, incluso otras funciones y definiciones de clases.

PHP no soporta la redefinición de funciones previamente declaradas.

Nota: Los nombres de funciones se pueden llamar con mayúsculas o minúsculas, aunque es una buena costumbre el llamar a las funciones tal y como aparecen en su definición.

Funciones internas (incorporadas):

PHP tiene incorporadas muchas funciones y construcciones. Existen también funciones que requieren extensiones específicas de PHP para que no fallen con un error fatal del tipo "undefined function" cuando son ejecutadas. Por ejemplo, para usar funciones de tratamiento de imágenes, tal como "imagecreatetruecolor()", se necesita compilar PHP con soporte para la librería **GD**; o para usar "mysql_connect()" se necesita compilar PHP con soporte para la librería **MySQL**.

Si se desea saber si determinada función está disponible para ser utilizada, se puede comprobar mediante "function_exists()", que nos devuelve un valor booleano indicando si el nombre que le hemos enviado como argumento es un nombre de función disponible por PHP.

Referencias a Variables:

Las Referencias en PHP son un medio para acceder al mismo contenido de una variable pero con diferentes nombres. No son como los punteros de C, sino que son alias en la tabla de símbolos. Hay que tener en cuenta, que en PHP el nombre de una variable y el contenido de una variable son diferentes, de manera que el mismo contenido, puede tener varios nombres. La analogía más cercana podría ser la de los archivos de Unix y sus nombres, dónde los nombres de las variables serían los directorios, y el contenido de las variables es el archivo en si. Las referencias también pueden ser pensadas como un enlace duro en un sistema de archivos Unix.

Las Referencias en PHP te permiten lograr que dos variables "apunten" al mismo contenido. Cuando haces algo como:

```
$a =& $b
```

significa que "$a" y "$b" apuntan a la misma variable.

Nota: "$a" y "$b" son completamente iguales, no es que "$a" esté apuntando a "$b" o viceversa, sino que tanto "$a" como "$b" apuntan al mismo lugar.

Más información sobre las referencias de variables en PHP:
http://www.php.net/manual/es/language.references.php

Otros elementos del lenguaje PHP:

El lenguaje PHP ha ido creciendo desde sus inicios, tanto que existen otros elementos y construcciones del lenguaje que por su extensión quedan fuera del estudio básico del mismo.

- **Clases, Interfaces y Objetos**
- **Espacios de Nombres (Namespaces)**
- **Excepciones (Exceptions)**

Se recomienda revisar la sintaxis y cómo trabajar con estos elementos del lenguaje PHP en la página Web del mismo: http://www.php.net

Referencias

Introducción al lenguaje PHP

http://www.php.net (Página oficial del lenguaje PHP con amplias explicaciones de la sintaxis del lenguaje, y muchos ejemplos en cada una de los apartados y funciones documentadas)
http://www.php.net/manual/es/tutorial.php (PHP - Un tutorial sencillo)

Integración de PHP y HTML

Programar en PHP

*** Consultar además estos enlaces con amplio material de estudio.

http://www.php.net/manual/es/tutorial.forms.php (PHP - Uso de Formularios)
http://www.php.net/manual/es/security.variables.php (PHP – Seguridad en Datos Enviados por el Usuario)

http://www.php.net/manual/es/language.oop5.php (PHP - Clases y Objetos)
http://www.php.net/manual/es/language.namespaces.php (PHP - Espacios de Nombres)
http://www.php.net/manual/es/language.exceptions.php (PHP - Excepciones)

http://www.php.net/manual/es/ref.mysql.php (PHP - Conexión con Bases de Datos MySQL)
http://www.php.net/manual/es/ref.odbc.php (PHP - Conexión con Bases de Datos de tipo ODBC)