Universidad Técnica Federico Santa María
Departamento de Electrónica
Programación de Sistemas

Programación con PHP

Profesor:
Sr. Agustín Gonzáles.
Alumno:
Christian Pelissier Q. 9821013-k

Fecha de Entrega: 30-10-2002

Índice

Introducción

El PHP acrónimo de (Hypertext Preprocessor) es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. El PHP se inició como una modificación a Perl escrita por Rasmus Lerdorf a finales de 1994. Su primer uso fue el de mantener un control sobre quien visitaba su curriculum en su web. En los siguientes tres años, se fue convirtiendo en lo que se conoce como PHP/FI 2.0. Esta forma de programar llego a muchos usuarios, pero el lenguaje no tomo el peso actual hasta que Zeev Surasky y Andi Gutmans le incluyeron nuevas características en 1997, que dio por resultado el PHP 3.0. La versión 4 es la más reciente.

Con PHP se puede hacer cualquier cosa que se pueda realizar con un script CGI, como el procesamiento de información en formularios, foros de discusión, manipulación de cookies y páginas dinámicas. Un sitio con páginas dinámicas es el que permite interactuar con el visitante, de modo que cada usuario que visita la página vea la información modificada para requisitos particulares. Las aplicaciones dinámicas para el Web son frecuentes en los sitios comerciales e-commerce, donde el contenido visualizado se genera de la información alcanzada en una base de datos u otra fuente externa.

Una de sus características más potentes es su soporte para gran cantidad de bases de datos. Entre su soporte pueden mencionarse MySQL, Oracle, ProgreSQL, entre otras. PHP también ofrece la integración con varias bibliotecas externas, que permiten que el desarrollador haga casi cualquier cosa desde generar documentos en pdf hasta analizar código XML.

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 puede hacer mucho más con PHP. Sus tags van incluidos dentro del código HTML. Su diseño elegante lo hace perceptiblemente más fácil de mantener y ponerse al día. Debido a su amplia distribución PHP esta perfectamente soportado por una gran comunidad de desarrolladores. Como producto de código abierto, PHP goza de la ayuda de un gran grupo de programadores, permitiendo que los fallos de funcionamiento se encuentren y se reparan rápidamente. El código se pone al día continuamente con mejoras y extensiones de lenguaje para ampliar sus capacidades.

Resumen

En este manual se explican los conceptos básicos del lenguaje PHP, seguidos de las instrucciones básicas para instalar este en un servidor Web Apache de una maquina que se encuentre corriendo el sistema operativo Linux.

Se señalan también los elementos de la sintaxis básica del lenguaje, como lo son los tags que determinan el inicio y termino del código, la separación de instrucciones, la manera en que se hacen los comentarios, como se definen las variables y sus características. Se muestran además algunos de los operadores que existen para interactuar con las variables como así también las estructuras de control y flujo que proporciona el lenguaje. Se muestra también como es el formato de salida y los comandos que la efectúan, además de todo lo que es el manejo de cadenas.

Se muestra la forma de como se define una función, así como el paso de parámetros a la misma, ya sea por referencia o por valor, también se muestran las distintas variantes par el retorno de los valores obtenidos. También se muestran las instrucciones para crear una librería de funciones y su respectiva invocación desde un código PHP cualesquiera.

Por último se describe el trabajo con formularios, las maneras que existen para enviar información de un formulario a un archivo que procesa esta, y la manera transparente y sencilla que entrega PHP para trabajar con estos.

En los anexos se describe lo que es la interacción de PHP con bases de datos, mas específicamente con la base de datos MySQL, como también distintas funciones para el manejo de archivos, además de algunas funciones de fecha y hora.

1- Conceptos básicos

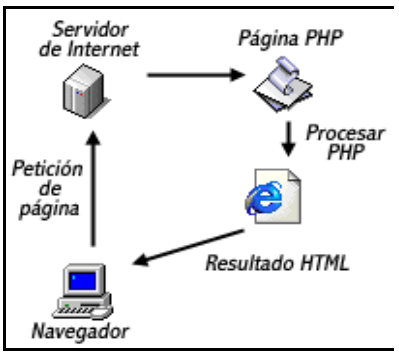
1.1- ¿Qué es PHP?

El lenguaje PHP es un lenguaje de programación de estilo clásico, es decir, es un lenguaje de programación con variables, sentencias condicionales, bucles y funciones, cercano a C o a JavaScript. No es un lenguaje de marcas como podría ser HTML, XML o WML.

PHP (acrónimo de "Hypertext Preprocessor") es un lenguaje "open source" interpretado de alto nivel embebido (introducido) en páginas HTML y ejecutado en el servidor. Es decir 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. Por ejemplo al acceder a una página escrita en PHP, el cliente solamente recibirá el resultado de la ejecución de esta en el servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido.

Gracias a que PHP se ejecuta en el servidor es posible acceder a los recursos que tenga el servidor como podría ser una base de datos.

A continuación se presenta un esquema del funcionamiento de PHP:



Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que el navegador lo soporte, es decir es independiente del navegador, pero sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

PHP puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, muchas variantes Unix (incluido 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 y muchos otros. PHP tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el estándar CGI, PHP puede usarse también como procesador CGI.

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 algunas de las cuales están soportadas actualmente: PostgreSQL, dBase, MySQL, Oracle, ODBC

1.2- Instrucciones Rápidas de Instalación (Versión Módulo de Apache)

En este capítulo se describirá el proceso de instalación, de un servidor web Apache con PHP, en una máquina con sistema operativo Linux o Unix, se supondrá que se tiene instalada un base de datos MySQL en el servidor.

Lo primero que se debe hacer es conseguir los paquetes necesarios, y que mejor para ello que dirigirse a las páginas web de los programas en cuestión:

Apache: http://www.apache.org

apache-1.3.x.tar.gz

PHP: http://www.php.net

php-3.0.x.tar.gz

Para poder realizar todo el proceso de instalación es necesario tener acceso como root a la máquina Linux en la cual se desea instalar estos programas.

Descargar todos los paquetes necesarios en un directorio en común para luego proceder a efectuar los siguientes pasos:

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-3.0.x.tar.gz
4.  tar xvf php-3.0.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/etc/httpd (dierctorio donde se instalara apache)
7.  cd ../php-3.0.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --prefix=/etc/httpd –activate
module=src/modules/php3/libphp3.a
13. make
14. make install
15. Edite su archivo de configuración de apache httpd.conf y añada:

            AddType application/x-httpd-php .php

  Puede elegir la extensión que desee aquí. .php es simplemente
sugerencia.

16. Utilice su método habitual para iniciar el servidor Apache (debe
detener (apachectrl start)
```

Para mayor información lea con mayor detención los readmes de instalación con los que cuentan los respectivos paquetes.

2- Referencias

2.1- Sintaxis básica

Para interpretar un archivo, PHP simplemente interpreta el texto del archivo hasta que encuentra uno de los caracteres especiales que delimitan el inicio de código PHP. El intérprete ejecuta entonces todo el código que encuentra, hasta que encuentra una etiqueta de fin de código, que le dice al intérprete que siga ignorando el código siguiente. Este mecanismo permite embeber código PHP dentro de HTML: todo lo que está fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Hay cuatro conjuntos de etiquetas que pueden ser usadas para denotar bloques de código PHP. De estas cuatro, sólo 2 (<?php. . .?> y <script language="php">. . .</script>) están siempre disponibles; el resto pueden ser configuradas en el fichero de php.ini para ser o no aceptadas por el intérprete. Mientras que el formato corto de etiquetas (short-form tags) y el estilo ASP (ASP-style tags) pueden ser convenientes, no son portables como la versión de formato largo de etiquetas. Además, si se pretende embeber código PHP en XML o XHTML, será obligatorio el uso del formato <?php. . .?> para la compatibilidad con XML.

Las etiquetas soportadas por PHP son:

```
1.  <?php echo("Manera clásica y más conveniente \n"); ?>

2.  <? echo ("esta es la más simple\n"); ?>
    <?= expresión ?> Esto es una abreviatura de "<? echo expresión ?>"

3.  <script languaje ="php">
        echo ("muchos editores (como FrontPage) no
              aceptan instrucciones de procesado");
    </script>

4.  <% echo ("Opcionalmente, puedes usar las etiquetas ASP"); %>
    <%= $variable; # Esto es una abreviatura de "<% echo . . ." %>
```

Ejemplo: Formas de escapar de HTML

2.2- Separación de instrucciones

La separación de instrucciones se hace de la misma manera que en C o Perl - terminando cada declaración con un punto y coma.

La etiqueta de fin de bloque (?>) implica el fin de la declaración, por lo tanto lo siguiente es equivalente:

```
<?php
    echo "este es un test";
?>
<?php echo "este es un test 2" ?>
```

2.3- Comentarios

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

```
<?php
    echo "hola este es un test"; // comentario de una línea como lo es
un comentario al estilo c++
     /* Este es un comentario
     Multilínea…… */
    echo "este es otro test test";
    echo "One Final Test"; # This is shell-style style comment

   <h1>Este es un <?php # echo "simple";?> ejemplo.</h1>
   <p>El header anterior dirá “Este es un ejemplo”.
    /*
    echo " Este es un test"; /* este comentario causara problemas */
    */
?>
```

Hay que tener cuidado con no anidar comentarios de estilo 'C', algo que puede ocurrir al comentar bloques largos de código.

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 // ?> será impreso; ?> sale del modo PHP, retornando al modo HTML, el comentario // no le influye.

2.4- Variables

Una variable es un contenedor de información, en el que se puede almacenar números enteros, números decimales, caracteres. El contenido de las variables se puede leer y se puede cambiar durante la ejecución de una página PHP.

En PHP todas las variables comienzan con el símbolo de peso ($) y no es necesario definir una variable antes de usarla. Tampoco tienen tipos, es decir que una misma variable puede contener un número y luego puede contener caracteres. El tipo de una variable lo decide PHP durante el tiempo de ejecución dependiendo del contexto en el que se utilice esa variable.

Se le asigna contenido con el signo igual (=). Para los nombres de las variables, PHP distingue entre mayúsculas y minúsculas, por lo que no es lo mismo $myvar que $Myvar, éstas son dos variables totalmente distintas.

A continuación se presenta un ejemplo, que muestra el uso de variables.

```
<?php
   //asignación de un valor a la variable
   $a = 1;   $b = 3.34;
   $c = "Hola Mundo";
   //imprime el valor asignado a las variables
   echo $a,"<br>",$b,"<br>",$c;
?>
```

En este ejemplo se han definido tres variables, $a, $b y $c, con la instrucción “echo” se logra imprimir el valor que contienen, insertando un salto de línea entre ellas.

Es importante destacar que existen 2 tipos de variables, las variables locales que solo pueden ser usadas dentro de funciones y las variables globales que tienen su ámbito de uso fuera de las funciones, se puede acceder a una variable global desde una función con la instrucción **global nombre_variable**.

2.4.1- Enteros

Los enteros se pueden especificar usando una de las siguientes sintaxis:

```
$x = 1234; # número decimal
$x = -123; # un número negativo
$x = 0123; # número octal (equivalente al 83 decimal)
$x = 0x12; # número hexadecimal (equivalente al 18 decimal)
```

2.4.2- Números en punto flotante

Los números en punto flotante ("double") se pueden especificar utilizando cualquiera de las siguientes sintaxis:

```
$x = 1.234;
$x = 1.2e3;
```

2.4.3- Cadenas

Las cadenas de caracteres se pueden especificar usando uno de dos tipos de delimitadores.

Si la cadena está encerrada entre dobles comillas ("), las variables que estén dentro de la cadena serán expandidas (sujetas a ciertas limitaciones de interpretación). Como en C y en Perl, el carácter de barra invertida ("\") se puede usar para especificar caracteres especiales:

Tabla: Caracteres protegidos

Secuencia	Significado
\n	Nueva línea
\r	Retorno de carro
\t	Tabulación horizontal
\\	Barra invertida
\$	Signo del dólar
\"	Comillas dobles

Las cadenas se pueden concatenar usando el operador '.' (Punto). Nótese que el operador '+' (suma) no sirve para esto.

Se puede acceder a los caracteres dentro de una cadena tratándola como un array de caracteres indexado numéricamente, usando una sintaxis similar a la de C, como se muestra en el ejemplo de a continuación.

Ejemplo: Algunos ejemplos de cadenas

```
<?php
//Asignando una cadena.
$str = "Esto es una cadena";
//Añadiendo a la cadena.
$str = $str . " con algo más de texto";
//Otra forma de añadir, incluye un carácter de nueva línea protegido.
$str .= " Y un carácter de nueva línea al final.\n";
// Esta cadena terminará siendo '<p>Número: 9</p>'
$num = 9;
$str = "<p>Número: $num</p>";
// Esta será '<p>Número: $num</p>'
$num = 9;
$str = '<p>Número: $num</p>';
// Obtener el primer carácter de una cadena
$str = 'Esto es una prueba.';
$first = $str[0];
// Obtener el último carácter de una cadena.
$str = 'Esto es aún una prueba.';
$last = $str[strlen($str)-1];
?>
```

2.4.4- Arrays

Los arrays actualmente actúan tanto como tablas hash (arrays asociativos) como arrays indexados (vectores).

Al igual que en otros lenguajes como C se tienen en PHP Arrays unidimensionales y bidimensionales. Se puede crear una array usando las funciones list() o array(), o se puede asignar el valor de cada elemento del array de manera explícita. También se puede crear un array simplemente añadiendo valores al array. Cuando se asigna un valor a una variable array usando corchetes vacíos, el valor se añadirá al final del array.

```
<?php
$a[0] = "hola a todos";
$a[1] = "como estan";
echo "$a[0]"."<br>";
echo ("$a[1]");
$a[] = "hola"; // $a[2] == "hola"
$a[] = "mundo"; // $a[3] == "mundo"
?>
```

```
<?php
 $dias = array (1=>'Lunes ',2=>'Martes ',
                        3=>'Miercoles ',4=>'Jueves ',
                        5=>'Viernes ', 6=>'Sabado ',
                        7=>'Domingo ');
  $mes = array (1=>'Enero',2=>'Febrero',
                        3=>'Marzo',4=>'Abril',
                        5=>'Mayo', 6=>'Junio',
                        7=>'Julio',8=>'Agosto',
                        9=>'Septiembre',10=>'Octubre',
                        11=>'Novimbre',4=>'Diciembre');
  $a= getdate();
echo "Hoy es: ".$dias[$a["wday"]].$a["mday"]." de ".$mes[$a["mon"]]." del
".$a["year"];
?>
```

2.4.5- Objetos

Para inicializar un objeto, se usa la sentencia new para instanciar el objeto a una variable.

```
class foo {
    function do_foo () {
        echo "Doing foo.";
    }
}
$bar = new foo;
$bar->do_foo();
```

2.5- Operadores

2.5.1- Aritméticos

Los operadores de PHP son muy parecidos a los de C y JavaScript, es por ello que al conocer estos lenguajes resultaran familiares y fáciles de reconocer.

Estos son los operadores que se pueden aplicar a las variables y constantes numéricas.

Operador	Nombre	Ejemplo	Descripción
+	Suma	5 + 6	Suma dos números
-	Resta	7 - 9	Resta dos números
*	Multiplicación	6 * 3	Multiplica dos números
/	División	4 / 8	Divide dos números
%	Módulo	7 % 2	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1
++	Suma 1	$a++	Suma 1 al contenido de una variable.
--	Resta 1	$a--	Resta 1 al contenido de una variable.

2.5.2- Comparación

Los operadores de comparación son usados para comparar valores y así poder tomar decisiones. A continuación se presenta una tabla donde se muestran los distintos tipos de operadores de comparación.

Operador	Nombre	Ejemplo	Devuelve cierto cuando:
==	Igual	$a == $b	$a es igual $b
!=	Distinto	$a != $b	$a es distinto $b
<	Menor que	$a < $b	$a es menor que $b
>	Mayor que	$a > $b	$a es mayor que $b
<=	Menor o igual	$a <= $b	$a es menor o igual que $b
>=	Mayor o igual	$a >= $b	$a es mayor o igual que $b

A continuación se presenta un ejemplo del uso de operaciones aritméticas:

```
<?php
   $a = 8; $b = 3;$c = 3;
   echo $a == $b,"<br>";
   echo $a != $b,"<br>";
   echo $a < $b,"<br>";
   echo $a >= $c,"<br>";
?>
```

2.5.3- Lógicos

Los operadores lógicos son usados para evaluar varias comparaciones, combinando los posibles valores de estas. A continuación se presenta una tabla donde se muestran los distintos tipos de operadores lógicos.

Operador	Nombre	Ejemplo	Devuelve cierto cuando:
&&	Y	(7>2) && (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
and	Y	(7>2) and (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
||	O	(7>2) || (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
or	O	(7>2) or (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
!	No	! (7>2)	Niega el valor de la expresión.

A continuación se presenta un ejemplo del uso de operaciones aritméticas:

```
<?php
   $a = 8; $b = 3; $c = 3;
   echo ($a == $b) && ($c > $b),"<br>";
   echo ($a == $b) || ($b == $c),"<br>";
   echo !($b <= $c),"<br>";
?>
```

2.6- Estructuras de Control

Todo archivo de comandos 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 con llaves. Un grupo de sentencias es también una sentencia. En este capítulo se describen los diferentes tipos de sentencias.

2.6.1- Sentencia if ... else

La sentencia **if** ejecuta una serie de instrucciones u otras dependiendo del resultado de evaluar una condición. Probablemente sea la instrucción más importante en cualquier lenguaje de programación, y esta estructurada igual que en C.

A continuación se presenta un ejemplo del uso de la sentencia if… else:

```
<?php
   $a = 8; $b = 3;
   if ($a < $b) {
     echo "a es menor que b";
   }
   else {
      echo "a no es menor que b";
   }
?>
```

2.6.2- Sentencia switch ... case

La sentencia **switch** es similar a una serie de sentencias IF en la misma expresión. En muchas ocasiones, se quiere comparar la misma variable (o expresión) con muchos valores diferentes, y ejecutar una parte de código distinta dependiendo de a qué valor es igual. Para ello sirve la sentencia **switch.**

A continuación se presenta un ejemplo del uso de la función switch:

```
<?php
   $posicion = "arriba";

   switch($posicion) {
      case "arriba":   // Bloque 1
         echo "La variable contiene";
         echo " el valor arriba";
         break;
      case "abajo":   // Bloque 2
         echo "La variable contiene";
         echo " el valor abajo";
         break;
      default:   // Bloque 3
         echo "La variable contiene otro valor";
         echo " distinto de arriba y abajo";
   }
?>
```

2.6.3- Bucles

En PHP se encuentran definidos básicamente los mismos bucles que en otros lenguajes como C, y se comportan de igual manera que estos.

- **while()**
- **do...while()**
- **for()**

Ejemplo:

```
Inicio<BR>
<?php
   $i=0;
   while ($i<10)
   {  echo "El valor de i es ", $i,"<br>";
      $i++;
   }
   for($i=0 ; $i<10 ; $i++)
   {
      echo "El valor de i es ", $i,"<br>";
   }
?>
Final<BR>
```

2.6- Salida

Hasta ahora se ha usado la instrucción **echo** para realizar salida a pantalla, pero esta instrucción es bastante limitada ya que no permite formatear la salida. Es por ello que se vera la instrucción **printf** que otorga mucha más potencia.

2.6.1- Sentencia printf

```
<?php
  printf(cadena formato, variable1, variable2...);
?>
```

La cadena de formateo indica cómo se han de representar las valores que posteriormente se indicaran. La principal ventaja es que además de poder formatear los valores de salida, permite intercalar texto entre ellos.

La cadena de formato puede incluir una seria de caracteres especiales que indican como formatear las variables que se incluyen en la instrucción.

Elemento	Tipo de variable
%s	Cadena de caracteres.
%d	Número sin decimales.
%f	Número con decimales.
%c	Carácter ASCII.
Aunque existen otros tipos, estos son los más importantes.	

A continuación se presenta un ejemplo de la utilización de la sentencia **printf:**

```
<?php
   $var="texto";
   $num=3;
   printf("Puede fácilmente intercalar <b>%s</b> con números <b>%d</b>
<br>",$var,$num);

   printf("<TABLE BORDER=1 CELLPADDING=20>");
   for ($i=0;$i<10;$i++)
       {
       printf("<tr><td>%10.d</td></tr>",$i);
       }
   printf("</table>");
?>
```

2.7- Manejo de cadenas

Dado el uso del lenguaje PHP el tratamiento de cadenas es muy importante, por ello existen bastantes funciones para el manejo de cadenas, a continuación se explicaran las más usadas.

1. ***strlen(cadena).*** Nos devuelve el número de caracteres de una cadena.
2. ***split(separador, cadena).*** Divide una cadena en varias usando un carácter separador.
3. ***sprintf(cadena de formato, var1, var2...).*** Formatea una cadena de texto al igual que printf pero el resultado es devuelto como una cadena.
4. ***substr(cadena, inicio, longitud).*** Devuelve una subcadena de otra, empezando por inicio y de longitud.
5. ***chop(cadena).*** Elimina los saltos de línea y los espacios finales de una cadena.
6. ***strpos(cadena1, cadena2).*** Busca la cadena2 dentro de cadena1 indicándonos la posición en la que se encuentra.
7. ***str_replace(cadena1, cadena2, texto).*** Reemplaza la cadena1 por la cadena2 en el texto.

A continuación se presenta un ejemplo con la utilización de cada una de las funciones señaladas anteriormente.

```
<?php
   echo strlen("12345"),"<br>";

   $palabras=split(" ","Esto es una prueba");
   for($i=0;$palabras[$i];$i++)
      echo $palabras[$i],"<br>";

   $resultado=sprintf("8x5 = %d <br>",8*5);
   echo $resultado,"<br>";

   echo substr("Devuelve una subcadena de otra",9,3),"<br><br>";
   if (chop("Cadena \n\n ") == "Cadena")
      echo "Iguales<br><br>";

   echo strpos("Busca la palabra dentro de la frase","palabra"),"<br><br>";

   echo str_replace("verde","rojo","La manzana es de color verde."),"<br>";
?>
```

3- Funciones

Al igual que en C el uso de funciones da la capacidad de agrupar varias instrucciones bajo un solo nombre y con ello poder llamar a estas varias veces desde diferentes sitios, ahorrándose así la necesidad de escribir de nuevo las respectivas líneas de código.

Una función se define con la siguiente sintaxis:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
    echo "Función de ejemplo.\n";
    return $retval;
}
```

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

En PHP3, las funciones deben definirse antes de que se referencien. En PHP4 no existe tal requerimiento.

A continuación se presenta un ejemplo con la utilización de las funciones.

```
<?php
   function media_aritmetica($a, $b){
      $media=($a+$b)/2;
      return $media;
      }
   echo media_aritmetica(4,6),"<br>";
   echo media_aritmetica(3242,524543),"<br>"; ?>
```

3.1- Parámetros de las funciones

La información puede suministrarse a las funciones mediante la lista de parámetros, una lista de variables y/o constantes separadas por comas.

PHP soporta pasar parámetros por valor (el comportamiento por defecto) y por referencia.

Pasar parámetros por valor y referencia

Por defecto, los parámetros de una función se pasan por valor (de manera que si cambias el valor del argumento dentro de la función, no se ve modificado fuera de ella). Si deseas permitir a una función modificar sus parámetros, debes pasarlos por referencia.

Para pasar un parámetro de una función por referencia, se debe anteponer un ampersand (&) al nombre del parámetro en la definición de la función:

Ejemplo:

```
function funci ($bar) {
    $bar .= ' y algo más.';
}
$str = 'Esto es una cadena, ';
funci ($str);
echo $str;    // Saca 'Esto es una cadena, '
funci (&$str);
echo $str;    // Saca 'Esto es una cadena, y algo más.'
```

3.2- Retorno de valores

Los valores se retornan usando la instrucción opcional **return**. Puede devolverse cualquier tipo de valor, incluyendo listas y objetos.

Ejemplo:

```
function square ($num) {
    return $num * $num;
}
echo square (4);   // saca '16'.
```

Es importante estacar que no se pueden devolver múltiples valores desde una función, pero un efecto similar se puede conseguir devolviendo una lista.

Ejemplo:

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

3.1- Librerías

El lenguaje PHP tiene definido el uso de librerías que permiten agrupar varias funciones y variables en un mismo fichero, de manera que luego incluir esta librería en distintas páginas y disponer de esas funciones fácilmente.

La instrucción para incluir una librería en una página es:

include("nombre de librería")

A continuación se procede a crear una librería (libreria1.php) a modo de ejemplo:

```
<?php
   function Cabecera_Pagina()
   {
?>
   <FONT SIZE="+1">
   Esta cabecera estará en todas sus páginas.
   </FONT><BR>
   <hr>
<?php
   }

   function Pie_Pagina()
   {
?>
   <hr>
   <FONT SIZE ="-1">Este es el pie de página.</FONT><BR>
   Autor: N.N
<? php
   }
?>
```

Ahora se procederá a crear una página que use la librería definida anteriormente.

Pagina1.php:

```
<html>
<body>
<?php include("libreria1.php") ?>
<?php Cabecera_Pagina(); ?>
Página 1
<BR><BR>
Aquí va el contenido <BR><BR>
más cosas...<BR><BR>
fin <BR><BR>
<?php Pie_Pagina(); ?>
</body>
</html>
```

4- Envío y recepción de datos con formularios

El lenguaje PHP proporciona una manera sencilla de manejar formularios, permitiendo de esta manera procesar la información que el usuario ha introducido.

Al diseñar un formulario se debe indicar la página PHP que procesará el formulario, así como en método por el que se le pasará la información a la página el cual es indicado en el atributo METHOD de la etiqueta FORM, existiendo dos métodos posibles los cuales son GET y POST.

La diferencia entre estos dos métodos radica en la forma de enviar los datos a la página, que procesa los datos obtenidos del formulario; mientras que el método GET envía los datos usando la URL, el método POST los envía por la entrada estándar STDIO.

A continuación se presenta un código simple de un formulario utilizando el método POST:

```
<html>
<body>
<H1>Ejemplo de procesado de formularios</H1>
<FORM ACTION="procesa2.php" METHOD="POST">
Introduzca su nombre:<INPUT TYPE="text" NAME="nombre"><BR>
Introduzca su apellidos:<INPUT TYPE="text" NAME="apellido"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
```

Al pulsar el botón Enviar el contenido de cuadro de texto es enviado a la página que se indica en el atributo ACTION de la etiqueta FORM.

PHP crea una variable por cada elemento del FORM, esta variable creada tiene el mismo nombre que el cuadro de texto de la página anterior y el valor que se haya introducido. En este ejemplo se ha creado una variable llamada $_GET["nombre"] con el valor que haya introducido el navegante.

A continuación se presenta el código que procesa la información recibida como resultado del envió del formulario:

procesa2.php:

```
<html>
<head>
   <title> procesa2.php </title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
El nombre que ha introducido es: <?php echo $_POST[“nombre”],"
",$ POST[“apellidos”] ?>
<br>
</body>
</html>
```

Anexos:

1- PHP y bases de datos (MySQL)

1.1- Creación de la base de datos

Para la realización de este tema sobre PHP con acceso a base de datos se ha elegido la base de datos MySQL por ser gratuita y por ser también la más empleada en entornos UNIX.

Para poder realizar los pasos siguientes es importante que el servidor en el cual se tienen alojadas las páginas debe tener la base de datos instalada, además debe tener habilitados los permisos para poder crearla.

El comando para crear una base de datos MySQL es el siguiente:

mysqladmin *-u root create base_datos*

Con este comando se consigue crear la una base de datos en el servidor de bases de datos del servidor en donde se desea alojar las paginas.

Una vez conseguido esto se deben crear las tablas en la base de datos, la descripción de las tablas contiene la estructura de la información que se almacenara en ellas. Para la creación se usara el lenguaje de consultas SQL el cual es común para todas las bases de datos relacionadas.

En este ejemplo creamos una tabla llamada prueba con 3 campos: un campo identificador, que nos servirá para identificar unívocamente una fila con el valor de dicho campo, otro campo con el nombre de una persona y por último un campo con el apellido de la persona.

Para crear la tabla puede usar la herramienta de administración de MySQL de su servidor web o puede escribir un fichero de texto con el contenido de la sentencia SQL equivalente y luego decirle al motor de base de datos que la ejecute con la siguiente instrucción:

mysql *-u root base_datos <prueba.sql*

prueba.sql

```
CREATE TABLE prueba (
ID_Prueba int(11) DEFAULT '0' NOT NULL auto_increment,
Nombre varchar(100),
Apellidos varchar(100),
PRIMARY KEY (ID_Prueba),
UNIQUE ID_Prueba (ID_Prueba)
);
```

1.2- Conexión a la base de datos

Una vez creada la base de datos en el servidor, el siguiente paso es conectarse a la misma desde una página PHP. Para ello PHP proporciona una serie de instrucciones para acceder a bases de datos MySQL.

```
<html>
<head>
   <title>Ejemplo de PHP</title>
</head>
<body>
<?php
function Conectarse()
{
   if (!($link=mysql_connect("localhost","usuario","Password")))
   {
      echo "Error conectando a la base de datos.";
      exit();
   }
   if (!mysql_select_db("base_datos",$link))
   {
      echo "Error seleccionando la base de datos.";
      exit();
   }
   return $link;
}
$link=Conectarse();
echo "Conexión con la base de datos conseguida.<br>";

mysql_close($link); //cierra la conexion
?>
</body>
</html>
```

Al ejecutar la instrucción **mysql_connect** se crea un vínculo entre la base de datos y la pagina PHP, este vínculo será usado posteriormente en las consultas que se hagan a la base de datos.

Finalmente, una vez que se ha terminado de usar el vínculo con la base de datos, se libera con la instrucción **mysql_close** para que la conexión no quede ocupada.

1.3- Consultas a la base de datos

Una vez que se ha logrado conectar con el servidor de bases de datos, ya se pueden realizar consultas a las tablas de la base de datos.

Para facilitar la programación se ha separado la función de conexión en una librería a parte, de tal manera que se incluirá en todas las páginas que accedan a la base de datos.

conex.php

```
<?php
function Conectarse()
{
   if (!($link=mysql_connect("localhost","usuario","Password")))
   {  echo "Error conectando a la base de datos.";
      exit();
   }
   if (!mysql_select_db("base_datos",$link))
   {  echo "Error seleccionando la base de datos.";
      exit();
   }
   return $link;
}
?>
```

Consulta.php

```
<html>
<head> <title>Ejemplo de PHP</title> </head>
<body>
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>
<?php
   include("conex.phtml");
   $link=Conectarse();
   $result=mysql_query("select * from prueba",$link);
?>
   <TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
  <TR><TD> Nombre</TD><TD> Apellidos </TD></TR>
<?php
   while($row = mysql_fetch_array($result))
   {
   printf("<tr><td> %s</td><td> %s </td></tr>",$row["Nombr
e"],$row["Apellidos"]);
   }
   mysql_free_result($result);
   mysql_close($link);
?>
</table>
</body>
</html>
```

En este ejemplo se han utilizado 3 instrucciones nuevas: **mysql_query**, **mysql_fetch_array** y **mysql_free_result**. La instrucción **mysql_query** es utilizada para hacer una consulta a la base de datos en el lenguaje de consultas SQL, con la instrucción **mysql_fetch_array** se extraen los datos de la consulta a un array y con **mysql_free_result** se libera la memoria usada en la consulta.

1.4- Inserción de registros

Hasta ahora se ha conseguido conectarse y hacer consultas a una base de datos, es por ello que ahora se presentara como introducir nuevos registros en la base de datos.

Para ello se usara un formulario y en el ACTION del FORM <FORM ACTION="programa.php"> se indicara la pagina php que procesara el formulario (esta página lo que hará será introducir los datos del formulario en la base de datos).

ejem07d.php

```
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>
<FORM ACTION="procesar.php">
<TABLE>
<TR>
   <TD>Nombre:</TD>
   <TD><INPUT TYPE="text" NAME="nombre" SIZE="20" MAXLENGTH="30"></TD>
</TR>
<TR>
   <TD>Apellidos:</TD>
   <TD><INPUT TYPE="text" NAME="apellidos" SIZE="20" MAXLENGTH="30"></TD>
</TR>
</TABLE>
<INPUT TYPE="submit" NAME="accion" VALUE="Grabar">
</FORM>
<hr>
<?php
   include("conex.php");
   $link=Conectarse();
   $result=mysql_query("select * from prueba",$link);
?>
   <TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
      <TR><TD> <B>Nombre</B></TD>
<TD> <B>Apellidos</B> </TD></TR>
<?php

   while($row = mysql_fetch_array($result)) {
      printf("<tr><td> %s</td> <td> %s </td></tr>",
$row["Nombre"], $row["Apellidos"]);
   }
   mysql_free_result($result);
   mysql_close($link);
?>
</table>
```

procesar.php

```
<?php
   include("conex.php");
   $link=Conectarse();
   mysql_query("insert into prueba (Nombre,Apellidos) values
('$nombre','$apellidos')",$link);

   header("Location: ejem07d.php");
?>
```

La primera página PHP ejem07d.php es un formulario que permite introducir los campos nombre y apellido para añadirlo a la base de datos, seguido de una consulta que muestra el contenido de la tabla prueba. El formulario llama a la pagina procesar.php que añadirá los datos a la tabla.

La segunda página procesar.php se conecta a la base de datos y añade un nuevo registro con la instrucción **insert** del lenguaje de base de datos SQL. Una vez el registro se ha añadido se vuelve a cargar la página ejem07d.php

1.5- Borrado de registros

Y finalmente, para cerrar el ciclo, queda el borrado de registros. El borrado de registros es uno de los procesos más sencillos.

Para indicar que elemento se va a borrar se ha usado un enlace a la página borra.php pasándole el Id_prueba de cada registro, de esta manera la página borra.php sabe que elemento de la tabla ha de borrar.

ejem07e.php

```
<html>
<body>
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>
<?php
   include("conex.php");
   $link=Conectarse();
   $result=mysql_query("select * from prueba",$link);
?>
   <TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
      <TR><TD> <B>Nombre</B></TD>
<TD> <B>Apellidos</B> </TD>
<TD> <B>Borrar</B> </TD></TR>
<?php
   while($row = mysql_fetch_array($result)) {
      printf("<tr><td> %s</td><td> %s </td><td><a
href=\"borra.php?id=%d\">Borra</a></td></tr>",
$row["Nombre"],$row["Apellidos"],$row["ID_Prueba"]);
   }
   mysql_free_result($result);
   mysql_close($link);
?>
</table>
</body>
</html>
```

borra.php

```
<?php
   include("conex.phtml");
   $link=Conectarse();
   mysql_query("delete from prueba where ID_Prueba =
$id",$link);

   header("Location: ejem07e.php");
?>
```

La página borra.php se conecta a la base de datos y borra el registro indicado en la variable $id que ha sido pasado desde la página ejem07e.php. Una vez el registro se ha borrado se vuelve a cargar la página ejem07e.php

2- Precedencia de Operadores

La precedencia de operadores especifica cómo se agrupan las expresiones. Por ejemplo, en la expresión 1 + 5 * 3, la respuesta es 16 y no 18 porque el operador de multiplicación ("*") tiene una mayor precedencia que el de adición ("+").

La siguiente tabla lista la precedencia de operadores, indicándose primero los de menor precedencia.

Asociatividad	**Operadores**
izquierda	,
izquierda	or
izquierda	xor
izquierda	and
derecha	print
izquierda	= += -= *= /= .= %= &= |= ^= ~= <<= >>=
izquierda	? :
izquierda izquierda	|| &&
izquierda	|
izquierda	^
izquierda	&
no asociativo	== != ===
no asociativo	< <= > >=
izquierda	<< >>
izquierda	+ - .
izquierda	* / %
derecha	! ~ ++ -- (int) (double) (string) (array) (object) @
derecha	[
no asociativo	new

3- Funciones de fecha y hora:

3.1- Comando date:

string date (string format [, int timestamp])

Devuelve una cadena formateada de acuerdo con la cadena de formato dada, utilizando el valor de timestamp dado o la hora local actual si no hay parámetro.

Se reconocen los siguientes caracteres en la cadena de formato:

a - "am" o "pm"
A - "AM" o "PM"
d - día del mes, dos dígitos con cero a la izquierda; es decir, de "01" a "31"
D - día de la semana, en texto, con tres letras; por ejemplo, "Fri"
F - mes, en texto, completo; por ejemplo, "January"
h - hora, de "01" a "12"
H - hora, de "00" a "23"
g - hour, sin ceros, de "1" a "12"
G - hour, sin ceros; de "0" a "23"
i - minutos; de "00" a "59"
j - día del mes sin cero inicial; de "1" a "31"
l ('L'minúscula) - día de la semana, en texto, completo; por ejemplo, "Friday"
L - "1" or "0", según si el año es bisiesto o no
m - mes; de "01" a "12"
n - mes sin cero inicial; de "1" a "12"
M - mes, en texto, 3 letras; por ejemplo, "Jan"
s - segundos; de "00" a "59"
S - sufijo ordinal en inglés, en texto, 2 caracteres; por ejemplo, "th", "nd"
t - número de días del mes dado; de "28" a "31"
U - segundos desde el valor de 'epoch'
w - día de la semana, en número, de "0" (domingo) a "6" (sábado)
Y - año, cuatro cifras; por ejemplo, "1999"
y - año, dos cifras; por ejemplo, "99"
z - día del año; de "0" a "365"
Z - diferencia horaria en segundos (de "-43200" a "43200")

Ejemplo:

```
<?php
print (date("l dS of F Y h:i:s A"));
print ("July 1, 2000 is on a " . date("l", mktime(0,0,0,7,1,2002)));
?>
```

3.2- Comando getdate:

array getdate (int timestamp)

Devuelve un array asociativo que contiene la información de fecha del valor timestamp como los siguientes elementos:

- **"seconds"** - segundos
- **"minutes"** - minutos
- **"hours"** - horas
- **"mday"** - día del mes
- **"wday"** - día de la semana, en número
- **"mon"** - mes, en número
- **"year"** - año, en número
- **"yday"** - día del año, en número; por ejemplo, "299"
- **"weekday"** - día de la semana, en texto, completo; por ejemplo, "Friday"
- **"month"** - mes, en texto, completo; por ejemplo, "January"

Ejemplo:

```
<?php
$a= getdate();
echo "Hoy: ".$a["weekday"]." ".$a["mday"]." de ".$a["month"]." del ".$a["year"];
?>
```

3.3- Comando gettimeofday:

array gettimeofday (void)

Es un interfaz para gettimeofday(2). Devuelve un array asociativo que contiene los datos devueltos por esta llamada al sistema.

- **"sec"** - segundos
- **"usec"** - microsegundos
- **"minuteswest"** - minutos al oeste de Greenwich
- **"dsttime"** - tipo de corrección dst

Ejemplo:

```
<?php
$a=gettimeofday ();
echo b["sec"];
?>
```

4- Funciones del sistema de ficheros

4.1- chmod

int chmod (string filename, int mode)

Trata de cambiar los permisos del fichero especificado por *filename* a los permisos dados por *mode*.

Ejemplo:

```
<?php
chmod( "/somedir/somefile", 0755 );  // octal; valor correcto de mode
?>
```

4.2- copy

int copy (string source, string dest)

Hace una copia de un fichero. Devuelve TRUE si la copia tiene éxito, y FALSE en otro caso.

Ejemplo:

```
<?php
$file=”pruebax.html”;
if (!copy($file, $file.'.bak'))
    {
    print("failed to copy $file...<br>\n");
    }
?>
```

4.3- fopen

int fopen (string filename, string mode [, int use_include_path])

Si **filename** comienza con "http://" (no es sensible a mayúsculas), se abre una conexión HTTP 1.0 hacia el servidor especificado y se devuelve un apuntador de fichero al comienzo del texto de respuesta. No maneja redirecciones HTTP, por eso se debe incluir una barra final cuando se trata de directorios.

Si **filename** comienza con "ftp://" (no es sensible a mayúsculas), se abre una conexión ftp hacia el servidor especificado y se devuelve un apuntador al fichero requerido. Si el servidor no soporta ftp en modo pasivo, esto fallará. Se pueden abrir fichero via ftp para leer o para escribir (pero no ambas cosas simultáneamente).

Si **filename** no comienza con nada de lo anterior, el fichero se abre del sistema de ficheros, y se devuelve un apuntador al fichero abierto.

Si el abrir el fichero falla, la función devuelve **FALSE**

mode puede ser cualquiera de los siguientes:

'r'	Abre para sólo lectura; sitúa el apuntador del fichero al comienzo del mismo.
'r+'	Abre para lectura y escritura; situa el apuntador del fichero al comienzo del fichero.
'w'	Abre para sólo escritura; sitúa el apuntador del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
'w+'	Abre el fichero para lectura y escritura; sitúa el apuntador del fichero al comienzo del fichero y trunca el fichero con longitud cero. Si el fichero no existe, trata de crearlo.
'a'	Abre sólo para escribir (añadir); sitúa el apuntador del fichero al final del mismo. Si el fichero no existe, trata de crearlo.
'a+'	Abre para lectura y escritura (añadiendo); sitúa el apuntador del fichero al final del mismo. Si el fichero no existe, trata de crearlo.

Ejemplo

```
$fp1 = fopen("/home/rasmus/file.txt", "r");
$fp2 = fopen("http://www.php.net/", "r");
$fp3 = fopen("ftp://user:password@example.com/", "w");
```

4.4- fread

string fread (int fp, int length)

fread() lee hasta *length* bytes del apuntador de fichero referenciado por *fp*. La lectura acaba cuando *length* bytes se han leido o se alcanza EOF, lo que ocurra primero.

Ejemplo:

```
<?php
// Mete el contenido de un fichero en una cadena
$filename = "/usr/local/something.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
?>
```

4.5- fscanf

mixed fscanf (int handle, string format [, string var1])

Ejemplo (lee un archivo que contiene 2 columnas nombre y apellido)

```
<?php
$fp = fopen ("datos.txt","r");
while ($userinfo = fscanf ($fp, "%s\t%s\n"))
    {
    list ($nombre, $apellido) = $userinfo;
    }
fclose($fp);
?>
```

4.6- fwrite

int fwrite (int fp, string string [, int length])

fwrite() escribe el contenido de *string* al fichero apuntado por *fp*. Si se da el argumento *length*, la escritura acaba antes de que *length* bytes sean escritos o se alcance el final de *string*, lo que ocurra primero.

Ejemplo:

```
<?php
 $filename="pruebax.html";
 $fd = fopen ($filename, "a+");
 $file = sprintf("Hola %s",$_POST["name"]);
  fwrite($fd,$file);
  echo $file;
fclose ($fd);
?>
```

4.7- Verificar el tipo de un fichero

Para saber el tipo de un fichero, en PHP se definen las siguientes funciones:

Función	***Sentencia***	***Retorno***
is_file	bool is_file (string filename)	Devuelve TRUE si el fichero nombrado existe y es un fichero regular.
is_dir	bool is_dir (string filename)	Devuelve TRUE si el nombre del fichero existe y es un directorio.
is_link	bool is_link (string filename)	Devuelve **TRUE** si el fichero indicado existe y es un enlace simbólico.
is_writable	bool is_writable (string filename)	Devuelve **TRUE** si el fichero indicado existe y se puede escribir en él. El argumento filename puede ser un directorio.
is_executable	bool is_executable (string filename)	Devuelve **TRUE** si el fichero indicado existe y es ejecutable.

A continuación se presentan otras funciones soportadas por PHP como mera referencia:

1. Funciones específicas de Apache
2. Soporte de las funciones COM para Windows
3. Funciones de Clases/Objetos
4. Funciones de pago electrónico
5. Funciones para dBase
6. Funciones con directorios
7. .NET functions
8. Funciones Forms Data Format (Formato de Datos de Formularios)
9. Funciones FTP
10. Funciones IMAP
11. Funciones matemáticas
12. Funciones de Red
13. ODBC functions
14. OpenSSL functions
15. Funciones Oracle
16. opciones e información de PHP
17. Funciones de PostgreSQL
18. Process Control Functions
19. Funciones de ejecución de programas
20. Funciones Semáforo y de memoria compartida
21. Shared Memory Functions
22. Funciones SNMP
23. Socket functions

Conclusión

A lo largo de este manual se han descrito y ejemplificado aspectos básicos para la programación con PHP, como también se han señalado algunos de los distintos tópicos que se pueden abordar con este lenguaje; lo que sin lugar a dudas sienta las bases para la generación de ideas para la realización de futuros proyectos de mayor envergadura como lo vendría siendo una aplicación en la cual exista una interacción a fondo con el procesamiento de formularios utilizando bases de datos, sean estas MySQL, Oracle u otra.

Bibliografía

Dadas las características del tema elegido la bibliografía se baso enteramente en páginas Web; las páginas utilizadas fueron:

- **http://www.php.net/manual/es/manual.php**
- http://otri.us.es/recursosPHP/manual/
- http://www.htmlpoint.com/php/guida/index.html