Programación en PHP/Texto Completo

De Wikilibros, la colección de libros de texto de contenido libre.
< Programación en PHP

Contenido

Obtener Apache Web Server y PHP

Obtener Apache y PHP por separado

Obtener Apache

El servidor de páginas web Apache se puede obtener de la página de descargas (http://httpd.apache.org/download.cgi) del sitio web del servidor HTTP Apache (http://httpd.apache.org/).

En el caso de Apache, existen dos líneas de desarrollo, la 1.x y la 2.x, siendo los complementos de una incompatibles con los de la otra.

Las versiones actuales del servidor son 1.3.33, 2.0.54 y 2.2.9.

Obtener PHP

PHP se puede obtener de la sección de descargas (http://www.php.net/downloads.php) del sitio web de PHP (http://www.php.net/).

En el caso de PHP, también hay dos líneas de desarrollo. PHP 4.x y PHP 5.x. En este caso, el grado de compatibilidad entre una y otra es bastante alto.

Las versiones actuales son 4.4.5 y 5.2.6.

Ambos programas pueden obtenerse de varias formas, siendo la más sencilla para usuarios noveles obtener el binario.

Obtener Apache y PHP juntos

Una solución para usuarios inexpertos es obtener en un paquete PHP y Apache. Uno de los paquetes que ofrecen esto es Easy PHP. Se puede descargar desde la web oficial (http://www.easyphp.org /telechargements.php3). Una vez instalado sólo hay que poner los archivos PHP en la carpeta www que está en la carpeta donde instalaste el servidor.

Hola mundo

[Volver al índice general]

Código

```php
<?php
$cadena = 'Hola mundo';
echo $cadena;
?>
```

Análisis

Esto es un código muy sencillo. Simplemente guarda la cadena de caracteres o string 'Hola mundo' en la variable $cadena y después se imprime por pantalla con el comando echo.

Nuevos conceptos

Inicio de un script en PHP

El ' <**?php** ' inicial indica que lo que sigue debería ser interpretado por PHP (por el intérprete de PHP), lo que significa que es parte del programa y no del HTML. Al final del documento, o cuando pasamos de nuevo a código HTML, se pone un ' **?**>' para indicar a PHP que ignore lo que venga a continuación. La etiqueta de cierre del código en PHP lleva implícito un ' **;** '.

Variables

Las variables son la base de cualquier lenguaje de programación. Sin ellas, no podríamos guardar información para utilizarla más adelante. PHP no sería capaz de hacer nada dinámico. Veamos, que las variables son necesarias y PHP las tiene.

Si tienes experiencia con otros lenguajes de programación, sabrás que en algunos de ellos es necesario declarar el tipo que va a tener una variable, es decir, el tipo de datos que será capaz de guardar. Son los lenguajes fuertemente tipados y en ellos necesitas saber el tipo de una variable antes de guardar cualquier cosa en ella. Son de este tipo lenguajes tan conocidos como C++ y Java. PHP, por otra parte, es un lenguaje debilmente tipado, porque el tipo de la variable corresponde al tipo del valor que está guardando *en este momento*. Puedes crear una variable para guardar una cadena de caracteres (string), asignar a esa variable un string, reemplazarlo luego por un número, y no habría problemas. Sin embargo, en C++ habría que haber hecho un *cast* (conversión explícita de tipos) o guardar ese número en otra variable distinta (del tipo correcto).

Todas las variables en PHP empiezan por '$' seguido de un identificador. Se diferencia entre mayúsculas y minúsculas, por lo que $wiki es diferente de $Wiki.

Para saber más de las variables, podéis mirar en el manual de PHP (http://www.php.net/manual /es/language.variables.php).

Print

La función **print** es clave para la salida de datos. Manda cualquier cosa que pongamos entre las comillas (o paréntesis) que siguen a **print** a la salida (la ventana del navegador). Una función similar podría se como **echo**, pero **print** permite al programador saber si la salida se ha realizado correctamente o no.

Un ejemplo de uso del **print** con comillas sería:

```
print 'Hola, mundo';
```

Al texto entre comillas se le trata como si fuera un string, por lo que podemos hacer uso de la concatenación (unir dos strings en uno). Por ejemplo:

```
print 'Hola, mundo';
```

y

```
print 'Hola' . ', ' . 'mundo';
```

Los puntos entre medias de las cadenas de caracteres o strings son el operador de concatenación. Los ejemplos anteriores tendrían el mismo efecto pues, en la segunda versión, los 3 strings se unen en uno solo antes de salir por pantalla.

La forma del comando **print** es una de las menos comunes. **Print** no es una función, sino una construcción interna. Las funciones, como veremos, tienen sus argumentos entre paréntesis.

```
miFuncion("Hola, mundo");
```

De todos modos, podemos usar **print** también con paréntesis, es decir:

```
print('Hola, mundo');
```

De cara a la optimización de código, cabe destacar, que se deben utilizar comillas simples ('), siempre que no vayas a devolver variables, pues con el uso de la doble comilla ("), el compilador debe analizar todo el contenido en busca de variables.

Es interesante hacer notar que **echo** se utiliza exactamente igual, por lo que si sustituimos en los ejemplos anteriores **print** por **echo** tendremos el mismo resultado.

Para saber más sobre **print** pulsa aquí (http://www.php.net/manual/es/function.print.php). Para **echo**, aquí (http://www.php.net/manual/es/function.echo.php).

El punto y coma y los espacios blancos

Otra parte imprescindible para este lenguaje, no así en otros como JavaScript ▸ (https://es.wikibooks.org/w/index.php?title=Programaci%C3%B3n_en_JavaScript/Desarrollo&action=edit) o ActionScript, es finalizar cada comando con un punto y coma ("**;**"). Sin él, el intérprete de PHP nos devolverá un error crítico y ni siquiera iniciará la ejecución del script.

Sin embargo PHP ignora completamente los espacios en blanco y los saltos de línea (salvo en las

definiciones de cadenas). Esto quiere decir que tanto podemos poner todo un script en una línea como sangrar cada línea tanto como queramos o dejar varias líneas en blanco.

Enlaces externos

- Hola Mundo en PHP (http://www.juarbo.com/hola-mundo-en-php/)

Bases del lenguaje

Es obligatorio colocar el código en las etiquetas `<?php ... ?>`, en un archivo .php, que se abre a través de un servidor web en HTTP.

Integración de HTML en PHP

```
<?php
 print '<html><h1>Hola</h1></html>';
?>
```

Integración de PHP en HTML

```
<html>
 <h1><?php print 'Hola'; ?></h1>
</html>
```

Comentarios

Los comentarios en php son muy similar a los comentarios en C, C++ y Pearl. Permite comentarios de una sola linea (de dos formas) y comentarios multilineas.

Los comentarios son escrituras fueras de la programacion, lo que significa que el interprete de PHP no ejecutara nada sobre esas lineas, las cuales tienen como unica utilidad especificar al programador en turno ciertas instrucciones, anotaciones o notas mentales de quien hizo el codigo.

Las etiquetas para comentarios en php son: "//", "#", **/*....*/** Ejemplo:

```php
<?php

//Comentario de una linea, la siguiente linea es leida como codigo de programacion
echo 'Linea de programacion';
#Comentario de una linea, tiene la misma utilidad que "//".
echo 'Linea de programacion';
/* Comentario multilinea
es capaz de abarcar muchas lineas,
posee una etiqueta de apertura y otra de cierre */
echo 'Linea de programacion';

?>
```

Variables

Introducción

En PHP las variables se representan con 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 válido tiene que empezar con una letra o un carácter de subrayado, seguido de cualquier número de letras, números y caracteres de subrayado.

Las variables en PHP se asignan, como en muchos otros lenguajes, con el símbolo igual (=).

Nota: *$this* es una variable especial que no puede ser asignada.

Un ejemplo:

```php
<?php
  $var = 12;
  $Var = 'Andrés';
  echo "$var $Var";  // Imprime: 12 Andrés
```

Inicialización y asignación de variables

PHP es un lenguaje no tipado. Esto significa que las variables no necesitan ser inicializadas y su tipo de dato no solo no precisa ser indicado, sino que éste puede cambiar. PHP simplemente sabe el tipo de dato que se utiliza en cada momento dependiendo del contexto en que se utilice.

Por ejemplo:

```php
<?php
  $var1 = 12;      // $var1 es un integer
  $var2 = 12.3;    // $var2 es un float
  $var2 = 'hola';  // $var2 es ahora un string
  $var2 = '13' + 1;// $var2 es ahora un integer
  echo $var2;      // Imprime: 14
```

No es necesario iniciar variables en PHP, sin embargo, es una muy buena práctica. Las variables no inicializadas tienen un valor predeterminado de acuerdo a su tipo dependiendo del contexto en el que son usadas.

Variables por valor y variables por referencia

Por defecto, en PHP las variables se asignan por valor. Esto significa que al asignar una expresión a una variable, el valor de la expresión es copiado a la variable.

Por ejemplo:

```php
<?php
  $var1 = 'hola';
  $var2 = $var1;         // Asigna por valor (por defecto)
  $var2 = 'adiós';       // Modifica $var2
  echo $var1 . " y " . $var2;  // Imprime: hola y adiós
```

Sin embargo, las variables también pueden asignarse por referencia. Esto significa que la nueva variable referencia a la variable original. En otras palabras, se convierte en un alias de (o simplemente apunta a) la variable original. Si una de las dos es cambiada, la otra también cambia su valor.

Para asignar por referencia, simplemente se antepone un signo ampersand (&) al comienzo de la variable cuyo valor se está asignando. Sólo las variables con nombre pueden ser asignadas por referencia.

Por ejemplo:

```php
<?php
  $var1 = 'hola';
  $var2 = &$var1;        // Asigna por referencia
  $var2 = 'adiós';       // Modifica $var1
  echo $var1 . " y " . $var2;  // Imprime: adiós y adiós
```

Fuente: Trubiso

Variables predefinidas

PHP proporciona una gran cantidad de variables predefinidas a cualquier script que se ejecute. Algunas de ellas son:

- **$GLOBALS** — Hace referencia a todas las variables disponibles en el ámbito global
- **$_SERVER** — Información del entorno del servidor y de ejecución
- **$_GET** — Variables HTTP GET
- **$_POST** — Variables HTTP POST
- **$_FILES** — Variables de Carga de Archivos HTTP
- **$_REQUEST** — Variables HTTP Request
- **$_SESSION** — Variables de sesión
- **$_ENV** — Variables de entorno
- **$_COOKIE** — Cookies HTTP

Ámbito de las variables

El ámbito de una variable es el contexto dentro del cual está definida. La mayor parte de las variables sólo tienen un ámbito simple, que también abarca los ficheros incluidos y requeridos. Por ejemplo:

```php
<?php
  $var1 = 15;
  include 'file.php';
```

La variable *$var1* estará disponible también al interior del script *file.php*.

En el caso de las funciones, al contrario que el lenguaje C, las variables definidas en un ámbito global no pueden ser accedidas dentro de un ámbito local (la función):

```php
<?php
  $var1 = 15;  // Ámbito global

  function prueba()
  {
      echo $var1;
  }
```

```
prueba();  // No produce salida
```

Variables globales accesibles

Para que una variable global sea accesible desde un ámbito local, como una función, se debe utilizar la palabra clave **global**:

```
<?php
  $var1 = 15;

  function prueba()
  {
      echo (global $var1);
  }

  prueba();  // Imprime 15
```

Otro método sería utilizar el array **$_GLOBALS**:

```
<?php
  $var1 = 15;

  function prueba()
  {
      echo $_GLOBALS['var1'];
  }

  prueba();  // Imprime 15
```

Variables estáticas

Una variable estática existe sólo en el ámbito local definido, pero no pierde su valor cuando la ejecución del programa abandona este ámbito.

Por ejemplo, el siguiente script:

```
<?php
  function prueba()
  {
      $var1 = 0;
      echo $var1;
      $var1++;
  }
```

Imprimiría *0* (cero) cada vez que se llamase a la función *prueba()*. Sin embargo, declarando una variable estática:

```
<?php
  function prueba()
  {
      static $var1 = 0;  // Variable estática
      echo $var1;
      $var1++;
  }
```

Imprimiría una secuencia (*0*, *1*, *2*, *3*, ...) cada vez que se llamase a la función *prueba()*, ya que el valor de $var1 se conserva estáticamente tras cada llamada a la función.

Las variables estáticas no se pueden redeclarar en la misma función (no se podría declarar otra vez *static $var1*).

Variables variables

Una variable variable es simplemente una variable cuyo nombre es el valor de otra variable. Por ejemplo:

```php
<?php
 $var1 = 'var2';
 $$var1 = 1;   // Se crea la variable $var2 = 1
 echo $var2;   // Imprime: 1
```

O también:

```php
<?php
 $var1 = 'var2';
 ${$var1} = 1;   // Se crea la variable $var2 = 1
 echo $var2;     // Imprime: 1
```

Cadenas

grupo Las **cadenas**, también llamadas strings (cadenas en inglés), son un tipo de datos que representan texto. Se llaman **cadenas** porque están formadas por caracteres únicos encadenados.

Como vimos en el ejemplo de "Hola, Mundo", podemos asignarlas a una variable e incluso combinarlas. Sin embargo es importante que conozcamos más sobre ellas y acerca de cómo funcionan.

Construcción

Una cadena se puede construir de tres maneras:

Comillas simples

Se basa en la delimitación del texto mediante comillas simples ('). Esta es la forma más sencilla de construir una cadena, pero también la más estricta: Todo lo que aparece en el interior es integramente la cadena.

```php
<?php
$nroPeras = 3;
$nroManzanas = 2;
$frase1 = 'Tengo $nroManzanas manzanas y $nroPeras peras.';
$frase2 = 'Tengo '.$nroManzanas.' manzanas y '.$nroPeras.' peras.';
echo $frase1 . ' - ' . $frase2;
?>
```

Tengo $nroManzanas manzanas y $nroPeras peras. - Tengo 2 manzanas y 3 peras.

Todo lo que introduzcamos aparecerá como tal y se almacenará como tal. En el caso de que necesitemos incluir una comilla simple en el texto, debemos *escaparla*. Para ello usaremos la barra invertida (\ ) antes de la comilla (\'). En el caso de que deseemos añadir una barra invertida al final de una cadena o antes de una comilla simple, deberemos escapar la propia barra invertida (\\ ). En cualquier otro caso, la barra invertida aparecerá normalmente.

```php
<?php
echo 'Ejemplo de \'cómo escapar varias comillas\',
de poner una barra invertida delante de una comilla \\\'
o de terminar la cadena con una barra invertida\\';
?>
```

Ejemplo de 'cómo escapar varias comillas', de poner una barra invertida delante de una comilla \' o de terminar la cadena con una barra invertida\

Como también vemos en este ejemplo, los saltos de línea naturales, es decir saltar la línea en la definición de la cadena, también se mantiene cuando la imprimimos en pantalla.

Comillas dobles

Cuando necesitamos incluir muchos valores guardados en variables dentro de una cadena, es tedioso tener que utilizar repetidamente el operador concatenador de cadenas (**.**) para unir variables con pedazos de cadena, como hemos visto en el primer ejemplo.

Para esto existe una forma de cadena más flexible, pero con el inconveniente de que hay más cosas que debemos tener en cuenta a la hora de insertar símbolos especiales. Podemos en este caso introducir variables dentro de la cadena y éstas serán interpretadas como su valor, no como su nombre:

```php
<?php
$nroPeras = 3;
$nroManzanas = 2;
$frase1 = "Tengo $nroManzanas manzanas y $nroPeras peras.";
$frase2 = "Tengo \$nroManzanas manzanas y \$nroPeras peras.";
echo $frase1 . " - " . $frase2;
?>
```

Tengo 2 manzanas y 3 peras. - Tengo $nroManzanas manzanas y $nroPeras peras.

El funcionamiento de la barra invertida es similar a las cadenas de comillas simples, pero se aplica, no sólo también al dólar (**$**), sino a una serie de letras que, cuando son escapadas, adquieren otro significado:

\n	Nueva línea
\r	Retorno de carro
\t	Tabulador
\$	Signo del dólar
\"	Comillas dobles
\\	Barra invertida
\###	Carácter representado por 1, 2 ó 3 cifras (#) en código octal
\x##	Carácter representado por 1 ó 2 cifras en código hexadecimal

Heredoc

Ésta es quizás la forma menos común de definir cadenas por su *extraña* sintaxis. No por ello debemos obviarla. Es una potente forma de tener toda la flexibilidad de las comillas dobles sin el problema de tener que escapar las comillas dobles.

Un ejemplo es el siguiente:

```php
<?php
$nroPeras = 3;
$nroManzanas = 2;
$frase = <<<FIN
Tengo:
\t$nroPeras peras
'''\t$nroManzanas manzanas'''
FIN
?>
```

Tengo:

3 peras
2 manzanas

En detalle, la primera línea contiene tres sígnos de *menor que* (<<<) y a continuación debe tener una serie de caracteres que finalizarán el texto. En nuestro ejemplo eran *FIN*, pero puede ser cualquier texto que no contenga un salto de línea. En las siguientes líneas ponemos el texto que queremos introducir en la cadena

tal como deseamos que aparezca, teniendo en cuenta la tabla superior y la interpretación de las variables. Cuando terminamos la cadena, en la siguiente línea sólo ponemos los caracteres que definimos al principio. No debe haber ningún otro carácter o espacio ni antes ni después de estos caracteres, de hecho, y como excepción, no es necesario el uso del punto y coma tras este texto finalizador.

Además, puesto que el texto finalizador es completamente personalizable podemos evitar errores. Si nuestro texto hubiera sido *Tengo*, por ejemplo, Al aparecer esta palabra al principio de una línea el interpretador de PHP hubiera entendido que ahí se acaba la cadena y habría producido un error al encontrar texto incongruente a continuación.

Para información más detallada sobre el uso y la sintaxis de las cadenas pulsa aquí (http://es.php.net/manual/es/language.types.string.php).

Operadores

Si tomamos de las matematicas que un operador es un símbolo matemático que indica que debe ser llevada a cabo una operación especificada[1] sobre un cierto número de operandos (número, función, vector, etc.), tendremos que en todo lenguaje de programación encontraremos muchos equivalentes y PHP no es la excepción y por su semejanza de C++ o Java tendremos que a un programador ya inducido en estos lenguajes no se le hará nada complicado.

Operadores Aritméticos:

```
-%a       //Negación
$a + $b   //Suma
$a - $b   //Resta
$a * $b   //Multiplicación
$a / $b   //División
$a % $b   //Resto de la división de $a entre $b
```

Operadores de incremento/decremento:

```
++$a      //Pre-incremento;    Incrementa $a en uno, y luego retorna $a.
$a++      //Post-incremento;   Retorna $a y luego incrementa en 1 a $a.
--$a      //Pre-decremento;    Decrementa $a en uno, y luego retorna $a.
$a--      //Post-decremento;   Retorna $a y luego decrementa en 1 a $a.
```

Operadores de Cadenas:

El único operador de cadenas que existen es el de concatenación, el punto. Pero no os asustéis, PHP dispone de toda una batería de funciones que os permitirán trabajar cómodamente con las cadenas.

```
$a = "Hola ";
$b = $a . "Mundo";   // Ahora $b contiene "Hola Mundo"
```

En este punto hay que hacer una distinción, la interpretación que hace PHP de las simples y dobles comillas. En el segundo caso PHP interpretará el contenido de la cadena.

```
$a = "Mundo";
echo 'Hola $a';   //Esto escribirá "Hola $a"
echo "Hola $a";   //Esto escribirá "Hola Mundo";
```

Operadores de Comparación:

```
$a <  $b   //$a menor que $b
$a >  $b   //$a mayor que $b
$a <= $b   //$a menor o igual que $b
$a >= $b   //$a mayor o igual que $b
$a <> $b   //$a es diferente de $b
$a == $b   //$a igual que $b
$a === $b  //$a es igual a $b, y son del mismo tipo de dato
$a !== $b  //$a no es igual a $b, o si no son del mismo tipo.
$a != $b   //$a distinto que $b
```

Operador Ternario:

Este es un operador algo particular pues [2] muchos lo ven como una estructura de control llamándolo "*if corto*" que nos permite simplificar una evaluación, pero cuidado, el utilizarlo de forma anidada puede darnos algunas sorpresas.

```
$accion = (true)?("comer"):("tomar"); //El valor de $accion será "comer"
$accion = (false)?("comer"):("tomar"); //El valor de $accion será "tomar"
/*
El siguiente código muestra como saber si una cadena es larga
o corta, para ello existen dos formas de hacerlo, con el operador
ternario y con el operador IF, ambos realizando lo mismo, veamos
NOTA: strlen devuelve un entero con el tamaño de caracteres de la cadena
*/

$cadena = "Esto es una cadena muy laaaaaarga";

//Con el Operador IF:

if (strlen($cadena)>3)
   echo "Es una cadena larga";
else
   echo "Es una cadena corta";

//Con el Operador Ternario:

echo (strlen($cadena)>3)?("Es una cadena larga"):"Es una cadena corta";
```

Operadores Lógicos:

```
$a AND $b  //Verdadero si ambos son verdadero
$a &&  $b  //Verdadero si ambos son verdadero
$a OR  $b  //Verdadero si alguno de los dos es verdadero
$a ||  $b  //Verdadero si alguno de los dos es verdadero
$a XOR $b  //Verdadero si sólo uno de los dos es verdadero
!$a        //Verdadero si $a es falso, y recíprocamente
```

Operadores de Asignación:

```
$a =  $b   //Asigna a $a el contenido de $b
$a = &$b   //Asigna a $a el contenido por referencia de $b
$a += $b   //Asigna a $a la suma de $b + $a
$a -= $b   //Asigna a $a la resta de $a - $b
$a *= $b   //Asigna a $a la multiplicación de $a por $b
$a /= $b   //Asigna a $a la división de $a entre $b
$a .= $b   //Asigna a $a la concatenación de $a seguida por $b
```

.

Los ejemplos

Ejemplo 1

En este ejemplo hacemos uso de 5 operadores básicos utilizados en las expresiones matemáticas. Son la base de todas las operaciones matemáticas y de string que se pueden llevar a cabo en PHP.

Estos 5 operadores matemáticos funcionan exactamente igual a como lo hacen en C++ o en Java. Son:

1. Suma (+)

2. Resta (-)
3. Multiplicación (*)
4. División (/)
5. Módulo (%) (el resto de la división por defecto)

Para asignar valores a variables utilizaremos =, que a diferencia de el significado matemático de "A **es igual que** B", en la mayoría de lenguajes de programación significa "A **toma el valor de** B". A este símbolo se le llama **operador de asignación**.

Código

```php
<?php
$x = 25;
$y = 10;
$z = $x + $y;
echo "$z\n";
$z = $x / $y;
echo "$z\n";
$z = $y * $y * $x;
echo "$z - 1250\n";
?>
```

Salida

```
35
2.5
1250
```

Referencias

1. Domingo Agustín Vázquez. «Diccionario de ciencias (http://books.google.es/books?id=_5-yHvJ61eQC)».
2. Mehdi Achour; Friedhelm Betz; Antony Dovgal; Nuno Lopes; Hannes Magnusson; Georg Richter; Damien Seguy; Jakub Vrana; Y muchos otros. «Manual de PHP (http://www.php.net/manual/es/language.operators.comparison.php#language.operators.comparison.ternary)».

Estructura if

La estructura **if** es muy sencilla.

Ejemplo Lógico:

```
Si el condicional se cumple o existe entonces ...
  Operaciones
```

Ejemplo PHP:

```php
if( 1 > 0 ) {
  echo "1 es mayor que 0";
}
```

Se puede complementar con **else** o **else if** este último se utiliza cuando queremos tener varios controles por si el primero no coincide, mira el siguiente condicional así secuencialmente hasta que entra. **else siempre va al final !**

Ejemplo Lógico:

```
Si el condicional se cumple o existe entonces ...
  Operaciones
Si no
  Otra operaciones
```

Ejemplo PHP:

```php
if( 1 > 0 ) {
  echo "1 es mayor que 0";
} else {
  echo "1 no es mayor a 0";
}
```

Estructura switch

Compara una variable con cada uno de los **case** previstos. Si coincide con uno de ellos ejecuta las instrucciones de su interior. De lo contrario ejecuta las instrucciones dentro de **default** (opcional). **break** termina con la ejecución de switch, sale del mismo.

```php
$value = 1;

switch ($value) {
  case 1:
    echo 'valor == 1';
    break;
  default:
    echo 'valor != 1';
    break;
}

// Con true
switch (true) {
  case $value == 1:
    echo 'valor == 1';
    break;
  default:
    echo 'valor != 1';
    break;
}

// Sintaxis alternativa
switch ($value):
  case 1:
    echo 'valor == 1';
    break;
  default:
    echo 'valor != 1';
endswitch;
```

Bucle while

while() ejecuta las sentencias en su interior mientras la condición sea verdadera.

```php
$valor = 0;

// mientras el valor sea menor que 3 imprime el valor
while ($valor < 3) {
  echo 'valor = ' . $valor . '<br />';
  $valor++; // incrementa 1
}

// Sintaxis alternativa
while ($valor < 3) :
  echo 'valor = ' . $valor . '<br />';
  $valor++; // incrementa 1
endwhile;
```

Funciones

Las funciones son rutinas creadas por el programador para realizar procesos que se repetirán o se usarán más de una vez. Las funciones pueden ser n-paramétricas (con $n \geq 0$), y estos parámetros pueden ser de entrada, de salida o de entrada y salida; aunque PHP no hace esta distinción del modo en que, por ejemplo, Ada (ver en Wikipedia), sí lo hace.

> En PHP, todas las funciones devuelven un valor. Aquellas que no devuelvan uno explícitamente, devuelven **NULL**, siguiendo el principio de que *todo en PHP es una expresión*.

Introducción

Esta sería una llamada a una función sin parámetros:

```
<?php
function Hola(){
$mensaje="Hola amigo";
echo $mensaje;
}
?>
<html>
<head>
<title>Mi Página</title>
</head>
<body>
<?php
Hola();
?>
</body>
</html>
```

El código superior ejecuta una llamada a la función

```
Hola()
```

, que imprimirá el texto "Hola amigo".

Funciones con parámetros.

Si necesitas una función a la cuál le puedas pasar parámetros, se escribiría del mismo modo, con el nombre de los parámetros entre los paréntesis, separados por comas. Para una función biparamétrica, el código podría ser algo como:

Función suma

```
<?php
function suma($a, $b){
```

```
  return $a + $b;
}

$a = 1; $b = 2;
echo "Sumemos $a + $b = " . suma($a, $b);
?>
```

Nótese que en este ejemplo hemos utilizado la palabra clave **return**. Esta palabra sirve para que la función devuelva valores al lugar en dónde se haya llamado.

A efectos prácticos digamos que la función se ejecuta en un entorno separado y se "sustituye" en tiempo de ejecución por el resultado devuelto por return en el contexto en el que se había llamado. En el ejemplo anterior, sumar($a, $b) === 3 (siendo === el operador de identidad).

Parámetros por defecto

En la definición formal de la función se pueden especificar parámetros por defecto, de tal modo que si no se pasa uno en el momento de llamarla, el parámetro toma un valor. Si este valor no se especifica, el intérprete devolverá un error en tiempo de ejecución por número de parámetros inválido.

Para especificarlos, se hace así:

Ejemplo de parámetros por defecto

```
function test($parametro = 'valor') {
  /* operaciones */
  return $parametro
}
```

Sobrecarga de funciones

Desde PHP4[Nota 1], se pueden *sobrecargar* funciones, esto es, definir funciones que aceptan distintos parámetros y puedan tener una lógica interna diferente.

Una función sobrecargada puede ser algo como:

Primer intento

```
function overload() {
  return array(1);
}

function overload($a) {
  return array(2, $a);
}

function overload($a, $b) {
  return array(3, array($a, $b));
}

function overload($a, $b, $c) {
  return array(4, array($a, $b, $c));
}
```

Falta decir que desde PHP5[Nota 1] las funciones sin parámetros especificados (con una lista vacía), pueden recibir cualquier número arbitrario de parámetros. La lista de funciones anteriores podría reescribirse como una sola, del siguiente modo:

Mejor aproximación

```
function not_overloaded() {
  return array(
```

func_num_args()PHP-Manual (http://es.php.net/func_num_args),

func_get_args()PHP-Manual (http://es.php.net/func_get_args)

```
);
}
```

Un ejemplo que muestra los parámetros utilizados, típico en los libros, es el siguiente:

function abc

```
function abc() {
```

```
    $array = func_get_args();
    $arrayCount = count($array);
    for($i = 0; $i < $arrayCount; $i++) {
      echo "Se utilizó como parámetro $i : {$array[$i]}";
    }
  }
```

Llamada a la función abc

```
abc('test', 'test2', 3, 4);
```

Resultado

```
Se utilizó como parámetro 0 : test
Se utilizó como parámetro 1 : test2
Se utilizó como parámetro 2 : 3
Se utilizó como parámetro 3 : 4
```

También se pueden crear funciones anónimas en tiempo de ejecución. Estas funciones son muy útiles en el lambda-cálculo, y se crean con **create_function**[PHP-Manual (http://es.php.net/create_function)], con un string en el segundo parámetro que sería el código PHP que debe de realizar la función. En PHP6 se introduce un nuevo estilo de funciones anónimas que permite además asignar funciones a variables, así como se hace en otros lenguajes como JavaScript.

De este modo, el siguiente código sería válido **en PHP6 o superior**.

Código experimental

```
 $a = function($b) { return $b; };
if($a(45) == 45) echo "Bien!"; // Se imprime 'Bien!'
```

Notas

1. El autor no recuerda desde qué versión mínima, pero seguro que en PHP5 en todas

Intermedio/Arrays

Introducción

Los arrays son una recolección de datos en una misma variable, por ejemplo un arreglo de los dias de la semana podría ser así:

```
$Dias = array('Lunes','Martes','Miércoles','Jueves','Viernes','Sábado','Domingo');
```

Ahora bien, si deseamos imprimir un contenido seria, por ejemplo:

```
echo "Hoy es ".$Dias[2]." y mañana sera ".$Dias[3];
```

Lo cual imprime:

```
Hoy es Miércoles y mañana sera Jueves
```

Asi pues podemos notar que el arreglo comienza a recorrerse a partir de la dirección 0, es decir Dias[0] corresponde a 'Lunes' y Dias[6] corresponde a 'Domingo'.

Si deseamos imprimir todos los días de la semana podríamos hacerlo en un ciclo utilizando una variable bandera "i", ejemplo:

```
for ($i=0;$i<=6;$i++)
 echo $Dias[$i]."," ;
```

Lo cual imprimiria:

```
Lunes,Martes,Miércoles,Jueves,Viernes,Sábado,Domingo
```

Existen varias formas de definir una variable como array (o un valor). Las más comunes son:

```
$variable = array(
            'indice' => 'valor',
            'valor', // Si se omite el índice el valor es el primer número natural no usado (de 0 a inf
            );
$variable[] = 'valor';
$variable['indice'] = 'valor';
```

Hay que recordar que al escribir dos veces un mismo índice del array, los datos se sobreescriben:

```
$array = array(
            'valor_del_indice_0'
        );
var_dump($array);
$array[0] = 'valor_nuevo';
var_dump($array);
```

Los ejemplos anteriores acceden mediante un índice a los elementos del array. PHP permite además acceder y modificar los arrays por métodos alternativos que pueden ser mucho más convenientes en determinadas ocasiones.

Utilizando arrays como pilas

Para añadir un elemento al final del array utilizamos la función array_push y para leer y eliminar el último elemento añadido utilizamos la función array_pop. Esto nos permite p.ej, utilizar un array como una pila para almacenar el estado intermedio de una operación:

```
<?php
  $pila = array();
  array_push($pila,"\n ");
  array_push($pila,"!");
  array_push($pila,"Mundo");
  array_push($pila,"Hola ");
  array_push($pila,"¡");
  print array_pop($pila);
  print array_pop($pila);
  print array_pop($pila);
  print array_pop($pila);
  print array_pop($pila);
?>
```

La salida será similar a:

```
¡Hola Mundo!
```

Arrays como diccionarios (tablas Hash)

Al igual que en las tablas de una base de datos muchas veces conviene indexar por una cadena de texto en vez de por un índice entero de forma que podemos buscar p.ej, el nombre de una persona a partir de su identificador fiscal. Supongamos que tenemos la siguiente tabla:

```
Identificador| Nombre
 Fiscal      |
=======================
000000001    | Fernando
000000002    | Marta
000000003    | Alfonso
```

Para representar la misma mediante un array PHP:

```
<?php
  $bbdd = array();
  $bbdd["000000001"]="Fernando";
  $bbdd["000000002"]="Marta";
  $bbdd["000000003"]="Alfonso";
  print $bbdd["000000003"];
?>
```

Un ejemplo ligeramente más complejo. Supongamos que la tabla tiene varias columnas:

```
Identificador| Nombre  |Apellido1|Apellido2|Edad|
 Fiscal      |         |         |         |    |
=============================================
000000001    | Fernando|Benito   |Alcantara| 53 |
000000002    | Marta   |Abenia   |Carrasco | 23 |
000000003    | Alfonso |Cordero  |Campo    | 45 |
```

Entonces puesto que nada impide que un elemento de un array sea a su vez otro array:

```php
<?php
  $bbdd = array();
  $bbdd["000000001"]=array("Fernando","Benito ","Alcantara","53");
  $bbdd["000000002"]=array("Marta   ","Abenia ","Carrasco ","23");
  $bbdd["000000003"]=array("Alfonso ","Cordero","Campo    ","45");
  print_r($bbdd["000000003"]);
?>
```

La salida del script será similar a:

```
Array
(
    [0] => Alfonso
    [1] => Cordero
    [2] => Campo
    [3] => 45
)
```

A la hora de crear/inicializar/definir el array podemos también utilizar la sintaxis:

```
array("clave1"=>"valor1","clave2"=>"valor2","clave3"=>"valor3",...)
```

lo cual nos permite mejorar el ejemplo anterior:

```php
<?php
  $bbdd = array();
  $bbdd["000000001"]=array("Nombre"=>"Fernando","Apellido1"=>"Benito ","Apellido2"=>"Alcantara","Edad"=>
  $bbdd["000000002"]=array("Nombre"=>"Marta   ","Apellido1"=>"Abenia ","Apellido2"=>"Carrasco ","Edad"=>
  $bbdd["000000003"]=array("Nombre"=>"Alfonso ","Apellido1"=>"Cordero","Apellido2"=>"Campo    ","Edad"=>
  print_r($bbdd["000000003"]);
  print("Edad:".$bbdd["000000003"]["Edad"]);
?>
```

La salida entonces será similar a:

```
Array
(
   [Nombre] => Alfonso
   [Apellido1] => Cordero
   [Apellido2] => Campo
   [Edad] => 45
)
Edad:45
```

Intermedio/Regexp

```
<?php
if (isset($_POST['muestra'])) {
    echo 'hola, '. htmlentities($_POST['nombre'])
    .', tu animal favorito es:'. htmlentities($_POST['animal']);
}
else {
?>
<from method="post" action="?">
¿cual es tu nombre?
<input type="text" name="nombre"/>
¿cual es tu animal favorito?
<select name="animal">
<option>perro</option>
<option>gato</option>
<option>ave</option>
</select>
<input type="submit" name="muestra" value="Siguiente">
</form>
<?php
}
?>
```

Intermedio/OOP

Php Orientado A Objeto

Cuando se trabaja en php orientado a objeto hay que tener en cuenta que se va a trabajar en la creacion de un objeto o mejor conocido en php como clases, donde le vamos a dar a este propiedades y metodos que describan al objeto y las acciones que realiza este. Al programar orientado a objeto hay que tener en cuenta que se tiene que pensar todo de forma abstracta hay que abstraer todo lo posible de este objeto para lograr crear un buen objeto.

Creacion de una Clase u objeto

Para crear una clase u objeto utilizando php se hace de la siguiente forma :

```
<?php
class NombreClase {
}
?>
```

En el ejemplo anterior solo se puede ver la como es la sintaxis para iniciar la creacion de un objeto , como se pueden dar cuenta se utiliza la palabra reservada class seguida del nombre de la clase y se abre una llave , de forma de que todo lo que esta dentro de esta llave pertenece a todo este objeto , dentro de estas llaves que abren y cierran van todo lo que es propiedades y metodos del objeto.

Propiedades de una clase

Las propiedades de las clases son los atributos del objeto como por ejemplo : el tamaño de una persona , el peso de una mesa. En php hay 3 niveles de acceso tanto para las propiedades como para los metodos los cuales son:

Public: Cuando le damos este nivel de acceso a un atributo , este puede ser accedido desde cualquier parte, asi sea dentro de su misma clase como desde otra.

Private: Cuando se le da el nivel de acceso private estamos quitandole la libertad de acceso a las demas que intenten acceder solo puede ser accedida desde su propia clase donde este fue definido.

Protected: Cuando le damos el nivel de acceso protegido sigue restringiendo la libertad de acceso a los demas que no sean de la clase donde se esta declarando o de sus subclases.

ahora veamos un ejemplo de como se colocaria la propiedad dentro del objeto en php.

```
<?php
class NombreClase {

  public $a;
  private $b;
  protected $c;

}
?>
```

Métodos de una clase

Como ya le habia nombrado antes los tipos de acceso tambien se utilizan en los metodo y son los mismos.

Para entender un poco que es el método de una clase u objeto , seria un ejemplo el de un objeto persona que tiene el metodo caminar , o sentarse o comer. Método seria todo lo que es capaz de realizar el objeto, en php su sintaxis utiliza la palabra reservada function, ahora veamos un ejemplo:

```
<?php
class NombreClase {

  public $a;
  private $b;
  protected $c;

  public function A($a) {
  }

  private function B() {
  }

  protected function C() {
  }

}
?>
```

Como pueden ver en el ejemplo se crearon 3 métodos dentro del objeto los cuales tienen cada uno de estos los distintos tipos de nivel de acceso , los metodos son mas que funciones pero las funciones que realiza dicho objeto.

Constructores

Cuando se trabaja programacion orientada a objeto se crea un constructor que este lo que hace es encargarse al momento de ser creado o instanciado el objeto inicializar algunas variables necesarias o basicas para el objeto , tambien puede ejecutar algun método de el mismo al momento de crearse todo esto es lo que nos permite hacer el constructor.

El constructor tiene la estructura de una funcion pero ya que esta dentro de un objeto este es un metodo con la diferencia de que ya tiene su nombre que es una palabra reservada **__construct** , ahora veamos un ejemplo de como seria un constructor:

```
<?php
class NombreClase {

  public $a;
  private $b;
  protected $c;

  function __construct() {
    $this->a = 'x';
    $this->b = 'w';
    $this->c = 'z';
  }

//...código
}
?>
```

De esta forma al momento de ser creado el objeto se van a inicializar las propiedades de el objeto como en el ejemplo donde en el objeto a la propiedad a se le asigna x , a la propiedad b se le asigna w y a la propiedad c se le asigna z, seguro se preguntan que es esa variable $this esta es lo que nos sirve para referirnos a que es en este objeto.

Al igual que hay un constructor existe un destructor que realiza todo el trabajo opuesto, que seria destruir todas las variables creadas.

Destructor

El destructor es utilizado para destruir todas las variables creadas en el objeto algo que no se utiliza mucho ya que php ya se encarga de liberar o eliminar todos los recursos utilizados al finalizar de ejecutar un script, sin embargo puede ser utilizado, su palabra reservada es **__destruct()**, vamos a ver un ejemplo a continuación:

```php
<?php
class NombreClase {

  public $a;
  private $b;
  protected $c;

  function __construct() {
    $this->a = 'x';
    $this->b = 'w';
    $this->c = 'z';
  }

  function __destruct() {
    echo "el atributo asignado " .$this->a . " ha sido eliminado por el destructor.";
  }

  //...código
}
?>
```

Este llamado al destructor se realiza al momento de finalizar de correr el script.

Funciones variadas

Math

Date

Manejo de ficheros

Trabajar con archivos es una parte importante en cualquier lenguaje de programación y PHP no es nada diferente. No importa cuales sean las razones por las que necesites manipular archivos, PHP los acomodará felizmente a través de una gran variedad de funciones. Deberías haber leído y comprendido los conceptos básicos de éste libro antes de seguir aquí.

fopen() y fclose()

fopen() es una de las funciones básicas para la manipulación de archivos. Abre un archivo en un cierto modo (que tu establecerás) y devuelve el recurso del puntero de archivo. Usando este recurso se puede leer y/o escribir en el archivo, antes de cerrarlo con la función fclose().

```
<?php

$puntero = fopen("datos.txt", "r"); // Abre el archivo sólo para lectura
fclose($puntero); // Cierra el archivo

?>
```

En el ejemplo anterior se puede observar que el archivo se abre sólo para lectura especificando "r" como el modo. Para ver una lista completa de los modos existentes para fopen() podes encontrarlos en la página Manual PHP (http://php.net/fopen).

Abriendo y cerrando el archivo es bien, pero para poder realizar operaciones útiles necesitas saber acerca de las funciones fread() (http://es.php.net/fread) y fwrite() (http://es.php.net/fwrite).

Cuando PHP finaliza la ejecución de un programa, todos los archivos abiertos son automáticamente cerrados. Asi que por lo tanto no es necesario cerrar un archivo luego de abrirlo y haber trabajado con el mísmo, pero a su vez, es considerado una buena practica de programación.

Lectura de archivos

La lectura de archivos se puede hacer de varias formas. Si tan solo lo que se necesita es el contenido del archivo, se puede usar la función file_get_contents() (http://es.php.net/file_get_contents). Si queremos cada una de las líneas del archivo en una matriz (array) se puede usar el comando file() (http://es.php.net/file). Para un control total sobre la lectura del archivo se recomienda utilizar fread() (http://es.php.net/fread).

Por lo general, estas funciones son intercambiables y cada una puede ser utilizada para cumplir con la función de cada una. Las primeras dos no requieren abrir el archivo con fopen() (http://es.php.net/fopen) o cerrarlo con fclose() (http://es.php.net/fclose). Éstas funciones son bastante útiles para operaciones rápidas realizadas de una sola vez. Si se planea realizar múltiples operaciones en un archivo es mejor utilizar fopen() (http://es.php.net/fopen) en conjunto con fread() (http://es.php.net/fread), fwrite() (http://es.php.net/fwrite) y fclose() (http://es.php.net/fclose) ya que es mucho más eficiente.

A continuación se muestra ejemplos utilizado las funciones mencionadas anteriormente.

Ejemplo usando file_get_contents() (http://es.php.net/file_get_contents):

```
<?php

$contenido = file_get_contents('datos.txt');
echo $contenido;
```

```
?>
```

El resultado es:

```
Soy el contenido de datos.txt
```

Esta función lee el archivo completo como una cadena (o string (http://es.php.net/string)) para luego ser manipulada como si fuese cualquier otra cadena.

Ejemplo usando file() (http://es.php.net/file):

```
<?php

$lineas = file('datos.txt');
foreach ($lineas as $numero => $linea) {
    $numero_de_linea = $numero + 1;
    echo "Linea $numero_de_linea: $linea";
}

?>
```

El resultado es:

```
Linea 1: Soy la primera linea de un archivo
Linea 2: Soy la segunda linea de un archivo
Linea 3: Si dijera que soy la cuarta linea del archivo, estaría mintiendo
```

Esta función lee el archivo completo como una matriz (o array (http://es.php.net/array)). Cada item en la matriz corresponde a una linea en el archivo.

Ejemplo usando fread() (http://es.php.net/fread):

```
<?php

$puntero = fopen("datos.txt", "r");
$cadena = fread($puntero, 64);
fclose($puntero);

echo $cadena;

?>
```

El resultado es:

```
Los primeros 64 bytes de datos.txt (siendo codificado en ASCII).
```

Esta función puede leer hasta una cantidad especificada de bytes del archivo y devolviéndolos como una cadena. Para la mayor parte, las primeras dos funciones son preferidas pero hay ocasiones en las que ésta función es necesaria.

Como puedes ver, con estas tres funciones se podrá leer fácilmente datos desde un archivo en la forma que sea conveniente para trabajar con los mismos. El siguiente ejemplo demuestra como estas tres funciones pueden se utilizadas para hacer los trabajos de las otras pero esto es opcional. Puede saltar y pasar hacia la sección Escritura si no está interesado.

```
<?php
```

```php
$archivo = "datos.txt";

function detectar_fin_de_linea($contenido) {
    if (false !== strpos($contenido, "\r\n")) return "\r\n";
    elseif (false !== strpos($contenido, "\r")) return "\r";
    else return "\n";
}

/* Esto es equivalente a file_get_contents($archivo) pero menos eficiente */
$puntero = fopen($archivo, "r");
$contenido = fread($puntero , filesize($archivo));
fclose($puntero);

/* Esto es equivalente a file($archivo) pero se requiere buscar el tipo de
   fin de línea. El sistema operativo Windows usa \r\n, Macintosh \r y Unix
   \n. file($archivo) detectará automaticamente el tipo de fin de línea
   mientras que fread()/file_get_contents() no lo harán. */
$fin_de_linea = detectar_fin_de_linea($contenido);
$contenido = file_get_contents($archivo);
$lineas = explode($fin_de_linea, $contenido);

/* Esto es equivalente a file_get_contents($archivo) */
$lineas= file($archivo);
$contenido = implode("\n", $lineas);

/* Esto es equivalente a fread($archivo, 64) si el archivo esta codificado en ASCII */
$contenido = file_get_contents($archivo);
$cadena = substr($contenido, 0, 64);

?>
```

Mysql

PHP es un lenguaje con capacidad de conexión con bases de datos. Puede comunicarse con distintos tipos, como MySQL, PostgreSQL, Interbase, Firebird, Informix, Oracle, MS SQL 7, Foxpro, Access, ADO, Sybase, FrontBase, DB2, SAP DB, SQLite y así cómo ODBC.

PHP y MySQL

En este apartado solo nos vamos a centrar en el uso de php con mysql, ya que sql es un lenguaje en particular, y muy generalizado. Normalmente, para la administración de bases de datos mysql, se usa PhpMyAdmin, una interfaz muy amigable con la que manejar las db. Así que comencemos:

Conexión

Lo primero que se debe realizar siempre, es la conexión de php con mysql. Para conectar con una base de datos, se usa la función mysql_connect(), donde deberemos indicar el servidor, el usuario y la clave. Aquí un ejemplo:

```
<?php
$dbhost = "localhost";
$dbuser = "wiki";
$dbpass = "books";
$conx = mysql_connect($dbhost, $dbuser, $dbpass);
?>
```

En las variables dbhost, dbuser, y dbpass, definimos el servidor, el usuario y la clave de nuestro servidor mysql. Y en la variable $conx, creamos la variable de conexión. A continuación, tenemos que seleccionar, dentro del servidor, la base de datos que queremos usar: Aquí un ejemplo:

```
<?php
$dbhost = "localhost";
$dbuser = "wiki";
$dbpass = "books";

$conx = mysql_connect($dbhost, $dbuser, $dbpass);

mysql_select_db("nuestra_db", $conx);

?>
```

Primera Consulta

Bien, llegados a este punto vamos a crear nuestra primera consulta. Vamos a contar con que el código de conexión y de selección de la DB lo tenemos escrito en el archivo conectar.php, y para ahorrarnos unas líneas, vamos a incluirlo en vez de reescribirlo. Para hacer una consulta, se usa la siguiente sintaxis: $sentencia = "SELECT [columna1, columna2] FROM nombre_tabla"; y después se ejecuta la consulta. Por último, se almacenan los datos en un array con la función mysql_fetch_array($nombredelapeticion). Aquí un ejemplo:

```
<?php
include("conectar.php");
```

```
$consulta = "SELECT usuario, clave FROM usuarios";
$peticion = mysql_query ($consulta, $conx);
$info = mysql_fetch_array($peticion);

//...y lo mostramos con un while, o como queramos..

?>
```

El código anterior nos mostraría todos los usuarios y sus claves de la tabla usuarios.

Nota: Si en vez de usuario y apellidos, que es seleccionar determinadas columnas de la tabla, usamos un asterisco (*), se seleccionarían todas las columnas de la tabla.

Bien, con eso lo que hemos hecho es seleccionar todos los registros existentes; pero, ¿y si queremos, usando el ejemplo anterior, la clave de determinado usuario? Para ello usaríamos WHERE. Aquí un ejemplo:

```
<?php
include("conectar.php");

$consulta = "SELECT * FROM usuarios WHERE usuario='pedro'";
$peticion = mysql_query ($consulta, $conx);
$info = mysql_fetch_array($peticion);

//...y lo mostramos con un while, o como queramos..
?>
```

La idea es simple. En pseudocódigo sería algo como: Selecciona todos los datos de el usuario pedro de la tabla usuarios. Se pueden ir añadiendo "filtros" como queramos: ...WHERE usuario='pedro' AND nombre='pedro picapiedra'...

Nota: Importante recordar usar siempre comillas simples al asignar valores. Es decir, es así: ...usuario='pedro'... no así: ...usuario="pedro".

Nota2: Para usar variables se usa igual: ...usuario='$miusuario'...

Y por último, hay más añadidos a la sentencia como ORDER BY nombre ASC (donde ordena alfabéticamente por el nombre en ASCendente) o LIMIT 0,30 la cual nos dará los resultados del registro 0 al 30 únicamente.

También podemos obtener en número de registros obtenidos con la función mysql_num_rows($nombredelapeticion). Que nos devolveria, por ejemplo: 5, si tenemos 5 registros en nuestra tabla.

Insertar datos

Para insertar datos, basta con indicar que columnas se van a insertar, y luego asignarle los valores a esas columnas. La sintaxis es:

$consulta = "INSERT INTO [nombre_tabla] ([columna1], [columna2],...,[columnaN]) VALUES ('$var1', '$var2',..., '$varN')";

Aquí un ejemplo (sin usar variables, son con texto):

```
<?php
include("conectar.php");

$consulta = "INSERT INTO usuarios (nombre, usuario, clave) VALUES ('pedro', 'pedro', 'clave')";
$peticion = mysql_query ($consulta, $conx);
?>
```

Actualizar Datos

Para actualizar, basta con indicar que columna se quiere cambiar, por el qué se quiere cambiar, y a quien se quiere cambiar. la sintaxis sería:

$consulta = "UPDATE [nombre_tabla] SET [columna1]='$valor1', [columna2]='$valor2' WHERE [columna3]='$valor3'";

Aquí un ejemplo (sin usar variables, son con texto):

```
<?php
include("conectar.php");

$consulta = "UPDATE usuarios SET nombre='paco', clave='123456' WHERE usuario='pedro'";
$peticion = mysql_query ($consulta, $conx);
?>
```

Otro ejemplo (con uso de variables):

Borrar Datos

Para borrar datos, basta con indicar a quien se quiere borrar o el que. La sintaxis sería:

$consulta = "DELETE [nombre_tabla] WHERE [columna1]='$valor1'";

Aquí un ejemplo (sin usar variables, son con texto):

```
<?php
include("conectar.php");

$consulta = "DELETE usuarios WHERE usuario='pedro'";
$peticion = mysql_query ($consulta, $conx);
?>
```

Esto nos habría borrado esa fila del usuario.

Avanzado/Graficos GD

Pendiente.. Escribir breve intro de que es GD.

Un nuevo concepto: Encabezados

Hasta ahora le hemos pedido a PHP que envíe texto al navegador, pero cuando trabajamos con imágenes, es necesario decirle al servidor web qué tipo de datos estamos enviando, para que a su vez, se lo notifique a los navegadores.

Para ello, utilizamos la función header (http://php.net/manual/es/function.header.php).

```php
<?php
header('Content-type: image/png');
?>
```

Esta función nos permite enviar mucha información al navegador; pero en esta ocasión, sólo nos interesa uno de los parámetros posibles: El parámetro "Content-type".

En el ejemplo anterior *Content-type*, o *Tipo de contenido*, nos permite decirle al navegador que la serie de datos que estamos enviando es una imágen y que es de tipo 'png'.

Manipulación de imágenes

Para cualquier imágen, vamos a requerir los siguientes bloques de código:

1. Crear / Abrir
2. Dibujar / Manipular / Escribir
3. Enviar encabezado
4. Desplegar imágen
5. Destruir imágen

Ya hablamos del encabezado, asi que ahora veremos cómo interactuar con la imágen.

Abrir, crear, desplegar y destruir

Para crear una imágen, se debe llamar a imagecreatetruecolor (http://php.net/manual/es/function.imagecreatetruecolor.php) o a imagecreate (http://www.php.net/manual/es/function.imagecreate.php).

Siempre que se manejen imágenes, es importante *destruirlas* al finalizar. Esto se logra con la función imagedestroy (http://php.net/manual/es/function.imagedestroy.php).

```php
<?php

$image = imagecreatetruecolor($width, $height);
imagealphablending($image, true);

header('Content-type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

En el ejemplo anterior, hay otra función a la cual prestarle atención: *imagePNG (http://php.net/manual /es/function.imagepng.php)*. Esta es la función responsable de desplegar nuestra imágen. Si se desea desplegar en otro formato de imágen (que no sea PNG); pueden utilizarse, por ejemplo, las funciones imageGIF (http://php.net/manual/es/function.imagegif.php), imageWBMP (http://php.net/manual /es/function.imagewbmp.php), imageJPEG (http://php.net/manual/es/function.imagejpeg.php), etcétera.

Para determinar los tipos de imágenes que soporta su sistema, puede llamar a la función imagetypes (http://php.net/manual/es/function.imagetypes.php).

Abrir una imágen ya existente puede presentar un inconveniente: la forma de abrirla depende del tipo de imágen del que se trata (por ejemplo: imageCreateFromGIF (http://php.net/manual /es/function.imagecreatefromgif.php), imageCreateFromJPEG (http://php.net/manual /es/function.imagecreatefromjpeg.php) o imageCreateFromPNG (http://php.net/manual /es/function.imagecreatefrompng.php)). Por eso, es útil tener una función que abra cualquier tipo de archivo.

El siguiente código, basado en un comentario en la página de PHP, trata de determinar el tipo de archivo del que se trata y utiliza la función correspondiente.

Se basa en la función getimagesize (http://php.net/manual/es/function.getimagesize.php), la cual regresa un arreglo del cual el elemento con índice 2 regresa un código dependiento del tipo de archivo. En caso de existir la función que abre ese tipo específico de archivo, lo abre. De lo contrario termina la ejecución.

```php
<?php
    // Esta funcion trata de determinar el tipo de archivo de imagen
    // y lo abre
    function imageCreateFromFile($filename)
    {
        static $image_creators;

        if (!isset($image_creators)) {
            $image_creators = array(
            1 => "imagecreatefromgif",
            2 => "imagecreatefromjpeg",
            3 => "imagecreatefrompng",
            6 => "imagecreatefromwbmp",
            16 => "imagecreatefromxbm"
            );
        }

        $image_size = getimagesize($filename);
        if (is_array($image_size)) {
            $file_type = $image_size[2];
            if (isset($image_creators[$file_type])) {
                $image_creator = $image_creators[$file_type];
                if (function_exists($image_creator)) {
                    return $image_creator($filename);
                } else {
                    die("imagecreatefrom: Function $image_creator doesn't exist!");
                }
            } else {
                die("imagecreatefrom: Image type is not supported!");
            }
        } else {
            die("imagecreatefrom: Not an array while calling getimagesize()!");
        }
    }
?>
```

Colores y canales alfa

En GD, los colores suelen estar representados en el modelo de color RGB (por las siglas en inglés Red, Green, Blue; es decir Rojo, Verde, Azul).

Sin embargo, hay un cuarto componente cuando elegimos un color (por ejemplo, con imagecreatetruecolor

(http://php.net/manual/es/function.imagecolorallocatealpha.php)): 'alfa'. El valor de *alfa* también es conocido como *transparencia*. En donde 255 es completamente transparente, y 0 es totalmente opaco.

En el siguiente ejemplo, se asignan 2 colores. Uno se utiliza para rellenar el fondo de la imágen y el otro para escribir texto en la imágen.

```
<?php
$image = imagecreatetruecolor($width, $height);
imagealphablending($image, true);
$background = imagecolorallocatealpha($image,230,230,230,0);
$foreground = imagecolorallocatealpha($image,0,0,0,0);
imagefill($image, 0,0,$background);
imagestring($image, 5, 0.05*$width, 0.05*$height, $string, $foreground);

header('Content-type: image/
imagepng($image);
imagedestroy($image);
?>
```

La función imagealphablending (http://php.net/manual/es/function.imagealphablending.php) establece el modo de mezcla de transparencia para una imágen. Esto es necesario si queremos que se mezclen los valores de transparencia en la imágen.

Líneas, círculos, rectángulos

Hay muchas funciones para dibujo, como lo son: imageFill, imageFilledRectangle, imageRectangle, imageLine, imageArc, etc..

Texto

```
<?php
$image = imageCreate($maxdimension,$maxdimension);
// elegimos colores..
$colorBG = imageColorAllocate($image, 192, 192, 192);
$colorTXT = imageColorAllocate($image, 0, 0, 255);
imageString($image, 5, 10, 40, "Imagen no encontrada", $colorTXT);
// creamos imagen entrelazada
imageInterlace($image, 1);
header("Content-type: image/png"); // Tipo de contenido
imagePNG($image);
imagedestroy($image);
exit();
?>
```

Tamaño de fuentes

Fuentes no predeterminadas

```
<?php
$string = "Hola, mundo!"; // Creamos una cadena
$stringlength = strlen($string); // Obtenemos su longitud
$font = imageloadfont("caveman.gdf");
$width = 1.1 * $stringlength * imagefontwidth($font);
$height = 1.1 * imagefontheight($font);
$image = imagecreatetruecolor($width, $height);
imagealphablending($image, true);
$background = imagecolorallocatealpha($image,230,230,230,0);
$foreground = imagecolorallocatealpha($image,0,0,0,0);
imagefill($image, 0,0,$background);
imagestring($image, $font, 0.05*$width, 0.05*$height, $string, $foreground);
```

```
header('Content-type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

Enlaces externos

- Devtrolls - Fuentes con licencia libre (http://www.devtrolls.com/gdf_fonts/fonts.html)

Avanzado/Graficos GD

Pendiente.. Escribir breve intro de que es GD.

Un nuevo concepto: Encabezados

Hasta ahora le hemos pedido a PHP que envíe texto al navegador, pero cuando trabajamos con imágenes, es necesario decirle al servidor web qué tipo de datos estamos enviando, para que a su vez, se lo notifique a los navegadores.

Para ello, utilizamos la función header (http://php.net/manual/es/function.header.php).

```php
<?php
header('Content-type: image/png');
?>
```

Esta función nos permite enviar mucha información al navegador; pero en esta ocasión, sólo nos interesa uno de los parámetros posibles: El parámetro "Content-type".

En el ejemplo anterior *Content-type*, o *Tipo de contenido*, nos permite decirle al navegador que la serie de datos que estamos enviando es una imágen y que es de tipo 'png'.

Manipulación de imágenes

Para cualquier imágen, vamos a requerir los siguientes bloques de código:

1. Crear / Abrir
2. Dibujar / Manipular / Escribir
3. Enviar encabezado
4. Desplegar imágen
5. Destruir imágen

Ya hablamos del encabezado, asi que ahora veremos cómo interactuar con la imágen.

Abrir, crear, desplegar y destruir

Para crear una imágen, se debe llamar a imagecreatetruecolor (http://php.net/manual/es/function.imagecreatetruecolor.php) o a imagecreate (http://www.php.net/manual/es/function.imagecreate.php).

Siempre que se manejen imágenes, es importante *destruirlas* al finalizar. Esto se logra con la función imagedestroy (http://php.net/manual/es/function.imagedestroy.php).

```php
<?php

$image = imagecreatetruecolor($width, $height);
imagealphablending($image, true);

header('Content-type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

En el ejemplo anterior, hay otra función a la cual prestarle atención: *imagePNG (http://php.net/manual /es/function.imagepng.php)*. Esta es la función responsable de desplegar nuestra imágen. Si se desea desplegar en otro formato de imágen (que no sea PNG); pueden utilizarse, por ejemplo, las funciones imageGIF (http://php.net/manual/es/function.imagegif.php), imageWBMP (http://php.net/manual /es/function.imagewbmp.php), imageJPEG (http://php.net/manual/es/function.imagejpeg.php), etcétera.

Para determinar los tipos de imágenes que soporta su sistema, puede llamar a la función imagetypes (http://php.net/manual/es/function.imagetypes.php).

Abrir una imágen ya existente puede presentar un inconveniente: la forma de abrirla depende del tipo de imágen del que se trata (por ejemplo: imageCreateFromGIF (http://php.net/manual /es/function.imagecreatefromgif.php), imageCreateFromJPEG (http://php.net/manual /es/function.imagecreatefromjpeg.php) o imageCreateFromPNG (http://php.net/manual /es/function.imagecreatefrompng.php)). Por eso, es útil tener una función que abra cualquier tipo de archivo.

El siguiente código, basado en un comentario en la página de PHP, trata de determinar el tipo de archivo del que se trata y utiliza la función correspondiente.

Se basa en la función getimagesize (http://php.net/manual/es/function.getimagesize.php), la cual regresa un arreglo del cual el elemento con índice 2 regresa un código dependiento del tipo de archivo. En caso de existir la función que abre ese tipo específico de archivo, lo abre. De lo contrario termina la ejecución.

```php
<?php
    // Esta funcion trata de determinar el tipo de archivo de imagen
    // y lo abre
    function imageCreateFromFile($filename)
    {
        static $image_creators;

        if (!isset($image_creators)) {
            $image_creators = array(
            1 => "imagecreatefromgif",
            2 => "imagecreatefromjpeg",
            3 => "imagecreatefrompng",
            6 => "imagecreatefromwbmp",
            16 => "imagecreatefromxbm"
            );
        }

        $image_size = getimagesize($filename);
        if (is_array($image_size)) {
            $file_type = $image_size[2];
            if (isset($image_creators[$file_type])) {
                $image_creator = $image_creators[$file_type];
                if (function_exists($image_creator)) {
                    return $image_creator($filename);
                } else {
                    die("imagecreatefrom: Function $image_creator doesn't exist!");
                }
            } else {
                die("imagecreatefrom: Image type is not supported!");
            }
        } else {
            die("imagecreatefrom: Not an array while calling getimagesize()!");
        }
    }
?>
```

Colores y canales alfa

En GD, los colores suelen estar representados en el modelo de color RGB (por las siglas en inglés Red, Green, Blue; es decir Rojo, Verde, Azul).

Sin embargo, hay un cuarto componente cuando elegimos un color (por ejemplo, con imagecreatetruecolor

(http://php.net/manual/es/function.imagecolorallocatealpha.php)): 'alfa'. El valor de *alfa* también es conocido como *transparencia*. En donde 255 es completamente transparente, y 0 es totalmente opaco.

En el siguiente ejemplo, se asignan 2 colores. Uno se utiliza para rellenar el fondo de la imágen y el otro para escribir texto en la imágen.

```
<?php
$image = imagecreatetruecolor($width, $height);
imagealphablending($image, true);
$background = imagecolorallocatealpha($image,230,230,230,0);
$foreground = imagecolorallocatealpha($image,0,0,0,0);
imagefill($image, 0,0,$background);
imagestring($image, 5, 0.05*$width, 0.05*$height, $string, $foreground);

header('Content-type: image/
imagepng($image);
imagedestroy($image);
?>
```

La función imagealphablending (http://php.net/manual/es/function.imagealphablending.php) establece el modo de mezcla de transparencia para una imágen. Esto es necesario si queremos que se mezclen los valores de transparencia en la imágen.

Líneas, círculos, rectángulos

Hay muchas funciones para dibujo, como lo son: imageFill, imageFilledRectangle, imageRectangle, imageLine, imageArc, etc..

Texto

```
<?php
$image = imageCreate($maxdimension,$maxdimension);
// elegimos colores..
$colorBG = imageColorAllocate($image, 192, 192, 192);
$colorTXT = imageColorAllocate($image, 0, 0, 255);
imageString($image, 5, 10, 40, "Imagen no encontrada", $colorTXT);
// creamos imagen entrelazada
imageInterlace($image, 1);
header("Content-type: image/png"); // Tipo de contenido
imagePNG($image);
imagedestroy($image);
exit();
?>
```

Tamaño de fuentes

Fuentes no predeterminadas

```
<?php
$string = "Hola, mundo!"; // Creamos una cadena
$stringlength = strlen($string); // Obtenemos su longitud
$font = imageloadfont("caveman.gdf");
$width = 1.1 * $stringlength * imagefontwidth($font);
$height = 1.1 * imagefontheight($font);
$image = imagecreatetruecolor($width, $height);
imagealphablending($image, true);
$background = imagecolorallocatealpha($image,230,230,230,0);
$foreground = imagecolorallocatealpha($image,0,0,0,0);
imagefill($image, 0,0,$background);
imagestring($image, $font, 0.05*$width, 0.05*$height, $string, $foreground);
```

```
header('Content-type: image/png');
imagepng($image);
imagedestroy($image);
?>
```

Enlaces externos

- Devtrolls - Fuentes con licencia libre (http://www.devtrolls.com/gdf_fonts/fonts.html)

Avanzado/La extensión SOAP

En La versión PHP >= 5 se puede trabajar directamente los metodos soap instanciando la clase para tal fin .

Para inicar trabajos con soap en PHP >= 5 se debe habilitar la extensión soap en el php.ini Nota: Si habilita la extensión soap o ya está habilitado no podría utilizar librerias como nusoap puesto que el nombre de las clases en las dos son iguales esto generaria un error.

Ejemplos/Calcular edad

Calcular edad con PHP

Introducción

Un cálculo muy solicitado en muchas ocasiones es el de calcular la edad. Es un calculo sencillo una vez tienes en cuenta algunos factores imprescindibles.

Si has nacido en 1980 y estamos en el 2000 tendrías 20 años siempre que hayas sobrepasado el día y el mes de tu nacimiento. Sabiendo esto generaremos dos if que restaran un año en el caso de no cumplir con la condición.

Código

```php
<?php
//fecha actual

$dia=date(j);
$mes=date(n);
$ano=date(Y);

//fecha de nacimiento

$dianaz=2;
$mesnaz=6;
$anonaz=1983;

//si el mes es el mismo pero el día inferior aun no ha cumplido años, le quitaremos un año al actual

if (($mesnaz == $mes) && ($dianaz > $dia)) {
$ano=($ano-1); }

//si el mes es superior al actual tampoco habrá cumplido años, por eso le quitamos un año al actual

if ($mesnaz > $mes) {
$ano=($ano-1);}

//ya no habría mas condiciones, ahora simplemente restamos los años y mostramos el resultado como su eda

$edad=($ano-$anonaz);

print $edad;

?>
```

Como veis es un código muy simple donde tendréis que sustituir las variables "$dianaz=2" "$mesnaz=6" y "$anonaz=1983" por la fecha de nacimiento a calcular para que os sea completamente útil.

Una manera más elegante y compacta de hacerlo puede ser:

```php
function CalculaEdad( $fecha ) {
    list($Y,$m,$d) = explode("-",$fecha);
    return( date("md") < $m.$d ? date("Y")-$Y-1 : date("Y")-$Y );
}
```

tg

Llamando a la función CalculaEdad() con la fecha en formato YYYY-mm-dd como parámetro, nos devuelve la edad de una persona nacida en esa fecha. Ej: CalculaEdad("1945-11-22");

Traspaso de variable

Aquí aprenderás a combinar PHP y HTML de manera eficiente y sencilla. Se tratará de explicar todo lo que ocurre de un paso a otro. Se presupone que si estás aquí es porque tienes instalado un servidor Apache con alguna versión de PHP. Si no, revisa la página inicial Programación en PHP.

Para iniciar una explicación detallada comenzaremos con un ejemplo (ejemplo.html):

```
<html>
<body>
   <form action="prueba.php" method="GET">
     <input type="text" name="prueba"> <input type="submit" value="Enviar">
   </form>
</body>
</html>
```

Esta es una sencilla página que posee un formulario que nos permitirá solicitar una información al usuario. Notesé que se usa el method="GET" es decir que va a pasar datos través de url del navegador, suena extraño pero en cuanto se finaliza el ejemplo se ve con claridad (Solo es necesario fijarse en la barra de direcciones del navegador). Es importante poner el name="prueba" ya que será el nombre de la variable o del dato que se pase de una página a otra.

Para hacerlo lo mas sencillo posible lo haremos de la siguiente forma (prueba.php):

```
<?php
if(!isset($_GET['prueba']))
{
 echo 'Variable "prueba" no definida.';
}
else
{
 echo $_GET['prueba'];
}
?>
```

Bien, lo que hace es simple y sencillo, comprobamos mediante la función isset (http://www.php.net/isset) que la variable "prueba" está definida, si lo está mostramos lo que escribió el usuario, en caso contrario mostramos un error que dice 'Variable "prueba" no definida.'. Debes tener presente que para que este ejemplo funcione debe realizarse bajo un servidor que use PHP, abre http://localhost/ejemplo.html para que funcione correctamente.

Utilidades/Intercambio de variables

Se pueden intercambiar variables, sus nombres, etc muy fácilmente:

Para hacer que a partir de una variable en un array se cree otra variable, se puede hacer esto:

```php
<?php
// Creamos una funcion que lo haga, es decir:
function crea_var_a_partir_de_matriz($array, $nombre)
{ // $array es la variable, entregada por valor y $nombre es el nombre de la variable que va a tener
    foreach($array as $key => $value)
    {
        global ${$nombre}.$key;
    }
}

$matriz = array();
$matriz['elem1'] = 'hola';
$matriz['elem2'] = 'mundo';
```

Avanzado/PHP, MySQL y la extension mysqli

Ejemplo de creación de una tabla con valores recuperados de una base de datos mysql:

```
<?php
/* primero generamos la conexion */
$conexion = mysql_connect('localhost', 'usuario', 'clave');
mysql_select_db('mibasededatos', $conexion);
/* generamos la peticion sql */
$result = mysql_query("SELECT nombre, email FROM agenda", $conexion);
echo "<table border = '1'> \n";
echo "<tr> \n";
echo "<td>'''Nombre'''</td> \n";
echo "<td>'''E-Mail'''</td> \n";
echo "</tr> \n";
/* almacenamos en un array $row, los valores que vamos recibiendo */
while ($row = mysql_fetch_array($result))
{
  echo "<tr> \n";
  echo "<td>$row[nombre]</td> \n";
  echo "<td>$row[email]</td> \n";
  echo "</tr> \n";
}
echo "</table> \n";
?>
```

Avanzado/XML en PHP 5

```
@$sCodUsu=$_GET['wcodusu'];
@$sClave=$_GET['wclave'];
@$sCurso=$_GET['wcurso'];
@$iCodAsi=$_GET['wcodasi'];
@$sCodConv=$_GET['wcodconv'];
@$sOrden=$_GET['worden'];
if ($sOrden=="") $sOrden="D";

//LLAMADA AL WEB SERVICE
$client = new SoapClient(null, array('location' => 'https://uxxi.cpd.ua.es:7779/UA-SI/WSS',
                                     'uri'      => 'http://UASI/WSS.wsdl',
                                     'encoding' => 'ISO-8859-1',
                                     'trace'    => 1) ); // el parámetro trace es para mostrar el XML in
$pLengua     = new SoapVar($sLengua , XSD_STRING, "string", "http://www.w3.org/2001/XMLSchema");
$pCodUsuario = new SoapVar($sCodUsu , XSD_STRING, "string", "http://www.w3.org/2001/XMLSchema");
$pClave      = new SoapVar($sClave , XSD_STRING, "string", "http://www.w3.org/2001/XMLSchema");
$pCurso      = new SoapVar($sCurso , XSD_STRING, "string", "http://www.w3.org/2001/XMLSchema");
$pCodAsi     = new SoapVar($iCodAsi , XSD_STRING, "string", "http://www.w3.org/2001/XMLSchema");
$pConv       = new SoapVar($sCodConv , XSD_STRING, "string", "http://www.w3.org/2001/XMLSchema");
$pOrden      = new SoapVar($sOrden , XSD_STRING, "string", "http://www.w3.org/2001/XMLSchema");
try
{
    $resultado = $client->wsfechaexamenesasi2(new SoapParam($pLengua, 'plengua'),
                                              new SoapParam($pCodUsuario, 'pcodusuario'),
                                              new SoapParam($pClave, 'pclave'),
                                              new SoapParam($pCurso, 'pcurso'),
                                              new SoapParam($pCodAsi, 'pcodasi'),
                                              new SoapParam($pConv, 'pconvocatoria'),
                                              new SoapParam($pOrden, 'porden') );

}
catch (SoapFault $exception)    //CONTROLAMOS LA EXCEPCIONES
{
    //Si el mensaje de error es nuestro, lo limpiamos para verlo mejor
    $delimitador = '@@';
    $e = $exception->faultstring;
    $p = strpos($e, $delimitador);
    if ($p !== false)
    {
        $q = strpos ($e, $delimitador,$p+2);
        $sError = substr($e, $p+2, $q-$p-2);
        echo "ERROR: ",$sError,"<BR>";
    }
    else
    {
        echo $e;
    }
    die();
}


// MOSTRAMOS LOS RESULTADOS
if ($sLengua == "C")
{
    $sTxtDia = "Día";
    $sTxtAsi = "Asignatura";
    $sTxtGrp = "Grupo";
    $sTxtConv = "Convocatoria";
    $sTxtObs = "Observaciones";
    $sTxtAulas = "Aulas";
}
else
{
    $sTxtDia = "Dia";
    $sTxtAsi = "Assignatura";
    $sTxtGrp = "Grup";
    $sTxtConv = "Convocatòria";
    $sTxtObs = "Observacions";
    $sTxtAulas = "Aules";
}

echo "<HTML><HEAD>\n";
echo "</HEAD><BODY>\n";
```

```
echo "<TABLE>\n";
echo "<tbody>\n";
echo "<TR>\n";
if ($sOrden=='D')
{
    echo " <TH>",$sTxtDia,"</TH>\n";
    echo " <TH>",$sTxtAsi,"</TH>\n";
    echo " <TH>",$sTxtGrp,"</TH>\n";
    echo " <TH>",$sTxtConv,"</TH>\n";
    echo " <TH>",$sTxtObs,"</TH>\n";
    echo " <TH>",$sTxtAulas,"</TH>\n";
}
elseif ($sOrden=='A')
{
    echo " <TH>",$sTxtAsi,"</TH>\n";
    echo " <TH>",$sTxtGrp,"</TH>\n";
    echo " <TH>",$sTxtConv,"</TH>\n";
    echo " <TH>",$sTxtDia,"</TH>\n";
    echo " <TH>",$sTxtObs,"</TH>\n";
    echo " <TH>",$sTxtAulas,"</TH>\n";
}
else
{
    echo " <TH>",$sTxtConv,"</TH>\n";
    echo " <TH>",$sTxtAsi,"</TH>\n";
    echo " <TH>",$sTxtGrp,"</TH>\n";
    echo " <TH>",$sTxtDia,"</TH>\n";
    echo " <TH>",$sTxtObs,"</TH>\n";
    echo " <TH>",$sTxtAulas,"</TH>\n";
}
echo "</TR>\n";

for ($i=0; $i<count($resultado->array);$i++)
{
    $valor = $resultado->array[$i];

    echo "<TR>\n";

    if ($sOrden=='D')
    {
        echo " <TD>",$valor->fecha,"</TD>\n";
        echo " <TD>",$valor->codasi,': ',$valor->nomasi,"</TD>\n";
        echo " <TD>",$valor->codgrp,"</TD>\n";
        echo " <TD>",$valor->codconv,': ',$valor->conv,"</TD>\n";
        echo " <TD>",$valor->observaciones,"</TD>\n";
        echo " <TD>",$valor->aulas,"</TD>\n";
    }
    elseif ($sOrden=='A')
    {
        echo " <TD>",$valor->codasi,': ',$valor->nomasi,"</TD>\n";
        echo " <TD>",$valor->codgrp,"</TD>\n";
        echo " <TD>",$valor->codconv,': ',$valor->conv,"</TD>\n";
        echo " <TD>",$valor->fecha,"</TD>\n";
        echo " <TD>",$valor->observaciones,"</TD>\n";
        echo " <TD>",$valor->aulas,"</TD>\n";
    }
    else
    {
        echo " <TD>",$valor->codconv,': ',$valor->conv,"</TD>\n";
        echo " <TD>",$valor->codasi,': ',$valor->nomasi,"</TD>\n";
        echo " <TD>",$valor->codgrp,"</TD>\n";
        echo " <TD>",$valor->fecha,"</TD>\n";
        echo " <TD>",$valor->observaciones,"</TD>\n";
        echo " <TD>",$valor->aulas,"</TD>\n";
    }

    echo "</TR>\n";
}
echo "</tbody>\n";
echo "</TABLE>\n";

echo "</BODY>\n";
echo "</HTML>\n";
?>
```

Esta categoría reúne todo un libro, cuya portada es:
Texto Completo **(https://es.wikibooks.org/w/index.php?title=Texto_Completo&action=edit&preload=Plantilla:Libro/%2BPortada&editintro=Plantilla:Libro/%21Portada)** ⊠ (https://es.wikibooks.org/w/index.php?title=Texto_Completo/Desarrollo&action=edit&preload=Plantilla:Libro/%2BDesarrollo&editintro=Plantilla:Libro/%21Desarrollo)

info (https://es.wikibooks.org/w/index.php?title=Texto_Completo/Info&action=edit&preload=Plantilla:Libro/%2BP%C3%A1gina&editintro=Plantilla:Libro/%21P%C3%A1gina) · *Índice* (https://es.wikibooks.org/w/index.php?title=Texto_Completo/%C3%8Dndice&action=edit&preload=Plantilla:Libro/%2BP%C3%A1gina&editintro=Plantilla:Libro/%21P%C3%A1gina) · *Introducción* (https://es.wikibooks.org/w/index.php?title=Texto_Completo/Introducci%C3%B3n&action=edit&preload=Plantilla:Libro/%2BP%C3%A1gina&editintro=Plantilla:Libro/%21P%C3%A1gina) · *Página de edición* (https://es.wikibooks.org/w/index.php?title=Texto_Completo/P%C3%A1gina_de_edici%C3%B3n&action=edit&preload=Plantilla:Libro/%2BEdici%C3%B3n&editintro=Plantilla:Libro/%21Edici%C3%B3n) · *Enlaces* (https://es.wikibooks.org/w/index.php?title=Texto_Completo/Enlaces&action=edit&preload=Plantilla:Libro/%2BP%C3%A1gina&editintro=Plantilla:Libro/%21P%C3%A1gina) · *Texto completo* (https://es.wikibooks.org/w/index.php?title=Texto_Completo/Texto_completo&action=edit&preload=Plantilla:Libro/%2BP%C3%A1gina&editintro=Plantilla:Libro/%21P%C3%A1gina)

Manuales de programación.

Categorías: Programación en PHP | Programación | Internet | Informática

---