PHP

Índice de contenido

Sintaxis básica

```php
<?php echo 'si se quiere mostrar documentos XHTML o XML, debe hacerse así'; ?>

<script language="php">
        echo 'algunos editores (como FrontPage) no les gusta
              las instruciones de proceso';
</script>
```

Comentarios

```php
<?php
    echo 'Esto es una prueba'; // Esto es un comentario estilo c++ de una sola línea
    /* Esto es un comentario multi-línea
       si! otra lína de comentarios */
    echo 'Un test final'; # Esto es un comentario estilo consola de una sola línea
?>
```

Tipos

PHP soporta **ocho tipos primitivos**.

Cuatro tipos escalares:

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

Dos tipos compuestos:

- array
- object

Y finalmente dos tipos especiales:

- resource
- NULL

Este manual introduce también algunos pseudo-tipos por razones de legibilidad:

- mixed
- number
- callback

> ***Nota:*** *Si desea chequear el tipo y valor de una expresión, use la función* ***var_dump()****.*
>
> *Podemos establecer o ver el tipo de una expresión con* ***settype()*** *y* ***gettype()***

Boolean

Para especificar un literal boolean, use alguna de las palabras clave TRUE o FALSE. Ambas son insensibles a mayúsculas y minúsculas.

```
<?php
      $foo = True; // asigna el valor TRUE a $foo
?>
```

Cuando se realizan conversiones a boolean, los siguientes valores son considerados **FALSE**:

- el boolean **FALSE** mismo
- el integer 0 (cero)
- el float 0.0 (cero)
- el valor string vacío, y el string "0"
- un array con cero elementos
- un object con cero variables miembro (sólo en PHP 4)
- el tipo especial NULL (incluyendo variables no definidas)
- objetos SimpleXML creados desde etiquetas vacías

Cualquier otro valor es considerado **TRUE** (incluyendo cualquier resource).

Moldes: (bool) y (boolean)

Integer

Los integers pueden ser especificados mediante notación decimal (base 10), hexadecimal (base 16), u octal (base 8), opcionalmente precedidos por un signo (- o +).

Para usar la notación octal, se antepone al número un *0* (cero). Para usar la notación hexadecimal, se antepone al número un *0x*.

```
<?php
      $a = 1234; // número decimal
      $a = -123; // un número negativo
      $a = 0123; // número octal (equivalente a 83 decimal)
      $a = 0x1A; // número hexadecimal (equivalente a 26 decimal)
?>
```

El tamaño de un integer depende de la plataforma. El tamaño de los integer puede ser determinado mediante la constante **PHP_INT_SIZE** y el valor máximo mediante la constante **PHP_INT_MAX** desde PHP 4.4.0 y PHP 5.0.5.

Moldes: (int) e (integer)

> **Nota:** Un valor tambien puede ser convertido a integer mediante la función **intval()**.

Float

Los números de punto flotante (también conocidos como "flotantes", "dobles" o "números reales") pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
<?php
	$a = 1.234;
	$b = 1.2e3;
	$c = 7E-10;
?>
```

El tamaño de un flotante depende de la plataforma, aunque un valor común consiste en un máximo de ~1.8e308 con una precisión de aproximadamente 14 dígitos decimales (lo que es un valor de 64 bits en formato IEEE).

Los numeros racionales que son representables exactamente como números de punto flotante en base 10, como 0.1 o 0.7, no tienen una representación exacta como números de punto flotante en base 2. Nunca confíe en resultados de números flotantes hasta el último dígito y nunca se comparan números de punto flotante para igualdad.

Molde: (float), (double) y (real)

> **Nota:** Un valor puede ser convertido a float mediante la función **floatval()**.

String

Un string es una serie de caracteres donde un caracter es lo mismo que un byte. Esto significa que PHP solo soporta el conjunto de 256 caracteres y por lo tanto no tiene soporte nativo Unicode.

Un string literal puede ser especificado de cuatro formas diferentes:

- comillas simples
- comillas dobles
- sintaxis heredoc
- sintaxis nowdoc (desde PHP 5.3.0)

Si un string está encerrado entre comillas dobles ("), PHP interpretará las variables y las sentencias de escape como caracteres especiales:

Caracteres escapados

Sentencia	Significado
\n	avance de línea (LF o 0x0A (10) en ASCII)
\r	retorno de carro (CR o 0x0D (13) en ASCII)
\t	tabulador horizontal (HT o 0x09 (9) en ASCII)
\v	tabulador vertical (VT o 0x0B (11) en ASCII) (desde PHP 5.2.5)
\f	avance de página (FF o 0x0C (12) en ASCII) (desde PHP 5.2.5)
\\	barra invertida
\$	signo del dólar
\"	comillas dobles

Los caracteres dentro de strings puede ser accedidos y modificados especificando la posición de caracter deseado (en base a la posición cero del primer caracter del string) empleando los corchetes de array, como en **$str[42]**. Los strings también pueden accederse utilizando llaves, como en **$str{42}**, para el mismo propósito.

Un valor puede convertirse a una string mediante el forzado *(string)* o la función **strval()**.

Molde: (string)

Nota: *Los strings pueden ser concatenados empleando el* ***operador '.'*** *(punto)*

Emplee las funciones ***ord()*** *y* ***chr()*** *para convertir entre código ASCII y caracteres.*

Existen numerosas funciones para el tratamiento de cadenas. Algunas de ellas son: ***strlen()***, ***trim ()***, ***strrev()***, ***strtolower()***, ***strtoupper()***, ***substr()***, ***strstr()***, ***strcmp()***

Array

Un array (matriz) en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves.

Un key puede ser un integer o bien un string. Un valor puede ser de cualquier tipo en PHP.

```php
<?php
	$arr = array("foo" => "bar", 12 => true);

	echo $arr["foo"]; // bar
	echo $arr[12];    // 1
?>
```

Array de dos dimensiones

```php
<?php
	$arr = array("somearray" => array(6 => 5, 13 => 9, "a" => 42));

	echo $arr["somearray"][6];    // 5
	echo $arr["somearray"][13];   // 9
	echo $arr["somearray"]["a"];  // 42
?>
```

Gestión de indices

```php
<?php
      // Este array es lo mismo que...
      array(5 => 43, 32, 56, "b" => 12);

      // ...este array
      array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Añadir y eliminar elementos

```php
<?php
      $arr = array(5 => 1, 12 => 2);

      $arr[] = 56;    // Esto es igual que $arr[13] = 56;
                      // en este punto del script

      $arr["x"] = 42; // Esto agrega un nuevo elemento a la
                      // array con la key "x"

      unset($arr[5]); // Esto elimina el elemento del array

      unset($arr);    // Esto elimina el array completamente
?>
```

Molde: (array)

Object

Para crear un nuevo object, utilice la declaración new para instanciar una clase:

```php
<?php
	class foo
	{
		function do_foo()
		{
			echo "Doing foo.";
		}
	}

	$bar = new foo;
	$bar->do_foo();
?>
```

Ejemplo de conversión a objecto

```php
<?php
	$obj = (object) 'ciao';
	echo $obj->scalar;  // muestra 'ciao'
?>
```

Molde: (object)

Recursos

Un valor tipo resource es una variable especial, que contiene una **referencia a un recurso externo**. Los recursos son creados y usados por funciones especiales

Dado que las variables resource contienen gestores especiales a archivos abiertos, conexiones con bases de datos, áreas de pintura de imágenes y cosas por el estilo, la conversión a tipo resource carece de sentido.

Nulo

El valor especial **NULL** representa una variable sin valor. **NULL** es el unico posible valor de tipo NULL.

Una variable es considerada null si:

- Se le ha asignado la constante **NULL**.
- No se ha establecido en un valor todavía.
- Se ha unset().

No hay más que un valor de tipo null, Y es el la palabra clave **NULL** entre mayúsculas y minúsculas.

```
<?php
	$var = NULL;
?>
```

Variables

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. Un nombre de variable válido tiene que **empezar con una letra o un carácter de subrayado** (underscore), seguido de cualquier número de letras, números y caracteres de subrayado

Las variables superglobales son:

- $GLOBALS
- **$_SERVER**
- $_GET
- $_POST
- **$_FILES**
- $_COOKIE
- $_SESSION
- **$_REQUEST**
- $_ENV
- $http_response_header

El array $http_response_header proporciona los encabezados de respuesta HTTP y es similar a la función **get_headers()**.

```
<?php
      file_get_contents ("http://example.com");
      print_r ($http_response_header);
?>
```

PHP también ofrece otra forma de asignar valores a las variables: **asignar por referencia**. Esto significa que la nueva variable simplemente referencia (en otras palabras, "se convierte en un **alias** de" ó "apunta a") la variable original. Los cambios a la nueva variable afectan a la original, y viceversa. Para asignar por referencia, simplemente se antepone un signo ampersand (**&**) al comienzo de la variable cuyo valor se está asignando (la variable fuente)

```
<?php
      $foo = 'Bob';               // Asigna el valor 'Bob' a $foo
      $bar = &$foo;               // Referenciar $foo vía $bar.

      $bar = "Mi nombre es $bar"; // Modificamos el original

      echo $bar;                  // Mi nombre es Bob
      echo $foo;                  // Mi nombre es Bob
?>
```

Ambito de las variables

```
<?php
$a = 1; /* ámbito global */

function test()
{
    echo $a; /* referencia a una variable del ámbito local */
}

test(); /* No imprimirá nada */
?>
```

En PHP, las variables globales deben ser declaradas globales dentro de la función si van a ser utilizadas dentro de dicha función.

```php
<?php
$a = 1;
$b = 2;

function Suma()
{
    global $a, $b;

    $b = $a + $b;
}

Suma();
echo $b;     /* Imprime 3 */
?>
```

La función suma también puede escribirse así:

```php
function Suma()
{
    $GLOBALS['b'] = $GLOBALS['a'] + $GLOBALS['b'];
}
```

> ***Nota:*** *Para mostrar el contenido de cualquier variable, use la función* ***print_r ()***

Constantes

Estas son las diferencias entre constantes y variables:

- Las constantes no llevan el signo dólar (*$*), como prefijo.
- Las constantes solo pueden ser definidas usando la función **define()**, y no por simple asignación.
- Las constantes pueden ser definidas y accedidas desde cualquier sitio sin importar las reglas de acceso de variables.
- Las constantes no pueden ser redefinidas o eliminadas una vez se han definido. Y
- Las constantes solo deberían contener valores escalares.

```php
<?php
      define ("CONSTANT", "Hola mundo.");
      echo CONSTANT; // muestra "Hola mundo."
      echo Constant; // muestra "Constant" y provoca un error.
?>
```

Nota: Las constantes pueden ser definidas usando la función **define()**

Operadores

Precedencia de operadores

Asociatividad	Operadores	Información adicional
no asociativo	clone new	clone and new
izquierda	[	array()
no asociativo	++ --	incremento/decremento
derecha	~ - (int) (float) (string) (array) (object) (bool) @	tipos
no asociativo	instanceof	tipos
derecha	!	lógico
izquierda	* / %	aritmética
izquierda	+ - .	aritmética y string
izquierda	<< >>	bit a bit
no asociativo	< <= > >= <>	comparación
no asociativo	== != === !==	comparación
izquierda	&	bit a bit y referencias
izquierda	^	bit a bit
izquierda	|	bit a bit
izquierda	&&	bit a bit
izquierda	||	lógico
izquierda	? :	ternario
derecha	= += -= *= /= .= %= &= |= ^= <<= >>= =>	asignación
izquierda	and	lógico
izquierda	xor	lógico
izquierda	or	lógico
izquierda	,	muchos usos

Operadores aritméticos

Ejemplo	Nombre	Resultado
-$a	Negación	Opuesto de *$a*.
$a + $b	Adición	Suma de *$a* y *$b*.
$a - $b	Sustracción	Diferencia de *$a* y *$b*.
$a * $b	Multiplicación	Producto de *$a* y *$b*.
$a / $b	División	Cociente de *$a* y *$b*.
$a % $b	Módulo	Resto de *$a* dividido por *$b*.

Operador de asignación

```php
<?php

$a = 3;
$a += 5; // establece $a en 8, como si se hubiera dicho: $a = $a + 5;
$b = "Hola ";
$b .= "ahí!"; // establece $b en "Hola ahí!", al igual que $b = $b . "ahí!";

?>
```

Operadores bit a bit

Ejemplo	Nombre	Resultado
`$a & $b`	And (y)	Los bits que están activos en ambos *$a* y *$b* son activados.
`$a | $b`	Or (o inclusivo)	Los bits que están activos ya sea en *$a* o en *$b* son activados.
`$a ^ $b`	Xor (o exclusivo)	Los bits que están activos en *$a* o en *$b*, pero no en ambos, son activados.

Ejemplo	Nombre	Resultado
`~ $a`	Not (no)	Los bits que están activos en $a son desactivados, y viceversa. Bits that are set in *$a* are not set, and vice versa.
`$a << $b`	Shift left(desplazamiento a izquierda)	Desplaza los bits de *$a*, *$b* pasos a la izquierda (cada paso quiere decir "multiplicar por dos").
`$a >> $b`	Shift right (desplazamiento a derecha)	Desplaza los bits de *$a*, *$b* pasos a la derecha (cada paso quiere decir "dividir por dos").

Operadores de comparación

Ejemplo	Nombre	Resultado
$a == $b	Igual	**TRUE** si *$a* es igual a *$b* después de la manipulación de tipos.
$a === $b	Idéntico	**TRUE** si *$a* es igual a *$b*, y son del mismo tipo. (a partir de PHP 4)
$a != $b	Diferente	**TRUE** si *$a* no es igual a *$b* después de la manipulación de tipos.
$a <> $b	Diferente	**TRUE** si *$a* no es igual a *$b* después de la manipulación de tipos.
$a !== $b	No idéntico	**TRUE** si *$a* no es igual a *$b*, o si no son del mismo tipo. (a partir de PHP 4)
$a < $b	Menor que	**TRUE** si *$a* es estrictamente menor que *$b*.
$a > $b	Mayor que	**TRUE** si *$a* es estrictamente mayor que *$b*.
$a <= $b	Menor o igual que	**TRUE** si *$a* es menor o igual que *$b*.
$a >= $b	Mayor o igual que	**TRUE** si *$a* es mayor o igual que *$b*.

Operadores de ejecución

PHP soporta un operador de ejecución: las comillas invertidas (`). ¡Note que estas no son las comillas sencillas! PHP intentará ejecutar el contenido entre las comillas invertidas como si se tratara de un comando del shell; la salida será retornada (es decir, no será simplemente volcada como salida; puede ser asignada a una variable). El uso del operador de comillas invertidas es idéntico al de shell_exec().

```
<?php
	$output = `ls -al`;
	echo "<pre>$output</pre>";
?>
```

> ***Nota:*** El operador de comillas invertidas se deshabilita cuando safe mode esta activado o shell_exec() esta desactivado.

Operadores de incremento/decremento

Ejemplo	Nombre	Efecto
++$a	Pre-incremento	Incrementa *$a* en uno, y luego retorna *$a*.
$a++	Post-incremento	Retorna *$a*, y luego incrementa *$a* en uno.
--$a	Pre-decremento	Decrementa *$a* en uno, luego retorna *$a*.
$a--	Post-decremento	Retorna *$a*, luego decrementa *$a* en uno.

Operadores lógicos

Ejemplo	Nombre	Resultado
$a and $b	And (y)	**TRUE** si tanto *$a* como *$b* son **TRUE**.
$a or $b	Or (o inclusivo)	**TRUE** si cualquiera de *$a* o *$b* es **TRUE**.
$a xor $b	Xor (o exclusivo)	**TRUE** si *$a* o *$b* es **TRUE**, pero no ambos.
! $a	Not (no)	**TRUE** si *$a* no es **TRUE**.

Ejemplo	Nombre	Resultado
$a && $b	And (y)	**TRUE** si tanto *$a* como *$b* son **TRUE**.
$a || $b	Or (o inclusivo)	**TRUE** si cualquiera de *$a* o *$b* es **TRUE**.

> ***Nota:*** La razón para tener las dos variaciones diferentes de los operadores "and" y "or" es que ellos operan con precedencias diferentes.

Operadores para strings

Existen dos operadores para datos tipo string. El primero es el operador de concatenación ('.'), el cual retorna el resultado de concatenar sus argumentos derecho e izquierdo. El segundo es el operador de asignación sobre concatenación ('.='), el cual añade el argumento del lado derecho al argumento en el lado izquierdo. Por favor consulte Operadores de asignación para más información.

```php
<?php
	$a = "Hello ";
	$b = $a . "World!"; // ahora $b contiene "Hello World!"

	$a = "Hello ";
	$a .= "World!";     // ahora $a contiene "Hello World!"
?>
```

Operadores para arrays

Ejemplo	Nombre	Resultado
$a + $b	Unión	Unión de *$a* y *$b*.
$a == $b	Igualdad	**TRUE** si *$a* i *$b* tienen las mismas parejas clave/valor.
$a === $b	Identidad	**TRUE** si *$a* y *$b* tienen las mismas parejas clave/valor en el mismo orden y de los mismos tipos.
$a != $b	Desigualdad	**TRUE** si *$a* no es igual a *$b*.
$a <> $b	Desigualdad	**TRUE** si *$a* no es igual a *$b*.
$a !== $b	No-identidad	**TRUE** si *$a* no es idéntica a *$b*.

Operadores de tipo

instanceof se utiliza para determinar si una variable de PHP es un objeto instanciado de una cierta clase:

```
<?php
class MyClass
{
}

class NotMyClass
{
}
$a = new MyClass;

var_dump($a instanceof MyClass);
var_dump($a instanceof NotMyClass);
?>
```

El resultado del ejemplo sería:

bool(true)
bool(false)

Estructuras de Control

Decisión

if/else

```php
<?php
/* ejemplo 1 */
if ($a == 5) {
    echo "a igual 5";
    echo "...";
}
elseif ($a == 6) {
    echo "a igual 6";
    echo "!!!";
}
else {
    echo "a no es 5 ni 6";
}

/* ejemplo 2 */
if ($a == 5):
    echo "a igual 5";
    echo "...";
elseif ($a == 6):
    echo "a igual 6";
    echo "!!!";
else:
    echo "a no es 5 ni 6";
endif;
?>
```

expresión ? :

```php
<?php
      $primero ? $segundo : $tercero
?>
```

switch

```php
<?php
/* ejemplo 1 */
switch ($i) {
    case 0:
        echo "i es igual a 0";
        break;
    case 1:
        echo "i es igual a 1";
        break;
    case 2:
        echo "i es igual a 2";
        break;
    default:
       echo "i no es igual a 0, 1 ni 2";
}

/* ejemplo 2 */
switch ($i):
    case 0:
        echo "i es igual a 0";
        break;
    case 1:
        echo "i es igual a 1";
        break;
    case 2:
        echo "i es igual a 2";
        break;
    default:
        echo "i no es igual a 0, 1 ni 2";
endswitch;
?>
```

Repetición

while

```
<?php
/* ejemplo 1 */

$i = 1;
while ($i <= 10) {
    echo $i++;  /* el valor presentado sería
                   $i antes del incremento
                   (post-incremento) */
}

/* ejemplo 2 */

$i = 1;
while ($i <= 10):
    echo $i;
    $i++;
endwhile;
?>
```

Hay una sola sintaxis para bucles ***do-while***:

```
<?php
$i = 0;
do {
    echo $i;
} while ($i > 0);
?>
```

for

```
<?php

for ($i = 1; $i <= 10; $i++) {
    echo $i;
```

```
}

?>
```

foreach

```
<?php
$arr = array("uno", "dos", "tres");

foreach ($arr as $value) {
    echo "Valor: $value<br />\n";
}

foreach ($arr as $key => $value) {
    echo "Clave: $key; Valor: $value<br />\n";
}
?>
```

Ruptura

break

break termina la ejecución de la estructura actual *for*, *foreach*, *while*, *do-while* o *switch*.

```
<?php

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "En 5<br />\n";
        break 1;  /* Sólo sale del switch. */
    case 10:
        echo "En 10; saliendo<br />\n";
        break 2;  /* Sale del switch y del while. */
    default:
        break;
    }
}
?>
```

continue

```
<?php
for ($i = 0; $i < 5; ++$i) {
    if ($i == 2)
        continue
    print "$i\n";
}
?>
```

return

Ejemplo #1 Uso de return()

```
<?php
function cuadrado($núm)
{
    return $núm * $núm;
}
echo cuadrado(4);   // imprime '16'.
?>
```

Ejemplo #2 Devolver una matriz para obtener múltiples valores

```
<?php
function números_pequeños()
{
    return array (0, 1, 2);
}
list ($cero, $uno, $dos) = números_pequeños();
?>
```

Ejemplo #3 Devolver una referencia desde una función

```
<?php
function &devolver_referencia()
{
    return $algunaref;
}
```

```php
$nuevaref =& devolver_referencia();
?>
```

Inclusión

require, require_once, include, include_once

La sentencia **include()** incluye y evalúa el archivo especificado.
require() es idéntico a include() excepto que en caso fallo parará el script mientras que include() sólo emitirá una advertencia (**E_WARNING**) lo cual permite continuar el script.
La sentencias **include_once()** y **require_once()** son idénticas a las anteriores excepto que PHP verificará si el archivo ya ha sido incluido y si es así, no se incluye de nuevo.

Ejemplo #1 Ejemplo básico de include()

vars.php

```php
<?php

$color = 'verde';
$fruta = 'manzana';

?>
```

test.php

```php
<?php
echo "Una $fruta $color"; // Una

include 'vars.php';

echo "Una $fruta $color"; // Una manzana verde
?>
```

Funciones

Funciones internas y funciones en extensiones

PHP se estandariza con muchas funciones y construcciones. También existen funciones que necesitan extensiones específicas de PHP compiladas, si no, aparecerán errores fatales "undefined function" ("función no definida"). Por ejemplo, para usar las funciones de image tales como imagecreatetruecolor(), PHP debe ser compilado con soporte para **GD**. O para usar mysql_connect(), PHP debe ser compilado con soporte para **MySQL**. Hay muchas funciones de núcleo que está incluidas en cada versión de PHP, tales como las funciones de string y de variable. Una llamada a **phpinfo()** o **get_loaded_extensions()** mostrará las extensiones que están cargadas en PHP.

Funciones definidas por el usuario

Las funciones no necesitan ser definidas antes de que se referencien .

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Función de ejemplo.\n";
    return $valordevuelto;
}
?>
```

En PHP es posible llamar a funciones recursivas. Sin embargo, evite las llamadas a funciones/métodos recursivos con más de 100-200 niveles de recursividad ya que pueden agotar la pila y causar la terminación del script actual.

```
<?php
function recursividad($a)
{
    if ($a < 20) {
        echo "$a\n";
        recursividad($a + 1);
    }
}
?>
```

PHP soporta argumentos pasados por valor (por defecto), pasados por referencia, y valores de argumentos predeterminados. Las Listas de argumentos de longitud variable también está soportadas .

Argumentos pasados por referencia

```php
<?php
function añadir_algo(&$cadena)
{
    $cadena .= 'y algo más.';
}
$cad = 'Esto es una cadena, ';
añadir_algo($cad);
echo $cad;    // imprime 'Esto es una cadena, y algo más.'
?>
```

Argumentos predeterminados

```php
<?php
function haceryogur($sabor, $tipo = "acidófilo")
{
    return "Hacer un tazón de yogur $tipo de $sabor.\n";
}

echo haceryogur("frambuesa");   // funciona como se esperaba
?>
```

Argumentos en cantidad variable

func_get_args es una función que devuelve una matriz que se compone de una lista de argumentos recibidos

```php
<?php
function foo()
{
    $númargs = func_num_args();
    echo "Número de argumentos: $númargs<br />\n";
    if ($númargs >= 2) {
        echo "El segundo argumento es: " . func_get_arg(1) . "<br />\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $númargs; $i++) {
        echo "El argumento $i es: " . $arg_list[$i] . "<br />\n";
    }
}
```

```
foo(1, 2, 3);
?>
```

El resultado del ejemplo sería:

```
Número de argumentos: 3
El segundo argumento es: 2
El argumento 0 es: 1
El argumento 1 es: 2
El argumento 2 es: 3
```

Funciones útiles

Para no hacer demasiado extensa esta sección he decidido hacer un breve resumen con las funciones más útiles. Las funciones para manejo de archivos o bases de datos no se ven aquí pues se verán posteriormente. No se muestra su forma de uso ni ejemplos. Para ello consultar el manual de PHP.

De tiempo y fechas

time - Devuelve el momento actual medido como el número de segundos desde la Época Unix (1 de Enero de 1970 00:00:00 GMT).
getdate - Devuelve un array asociativo que contiene la información de la fecha de *timestamp*, o el momento local actual si no se da *timestamp*.

Matemáticas

abs -- Valor absoluto
max — Encontrar el valor más alto
min — Encontrar el valor más bajo
pi — Obtener valor de pi
sqrt — Raíz cuadrada
rand — Genera un número entero aleatorio

base_convert — Convertir un número entre bases arbitrarias
bindec — Binario a decimal
decbin — Decimal a binario
dechex — Decimal a hexadecimal
decoct — Decimal a octal
hexdec — Hexadecimal a decimal

octdec — Octal a decimal

round — Redondea un float
ceil — Redondear fracciones hacia arriba
floor — Redondear fracciones hacia abajo

sin — Seno
cos — Coseno
tan — Tangente

deg2rad — Convierte el número en grados a su equivalente en radianes
rad2deg — Convierte el número en radianes a su equivalente en grados

exp — Calcula la exponencial de e
log10 — Logaritmo en base 10
log — Logaritmo natural
pow — Expresión exponencial

Cadenas de texto
echo — Muestra una o más cadenas
chr — Devuelve un caracter específico
ord — devuelve el valor ASCII de una caracter
implode — Une elementos de un array en una cadena
join — Alias de implode
explode — Divide una cadena en varias cadenas

trim — Elimina espacio en blanco (u otro tipo de caracteres) del inicio y el final de la cadena
ltrim — Retira espacios en blanco (u otros caracteres) del inicio de un string
rtrim — Retira los espacios en blanco (u otros caracteres) del final de un string

str_replace — Reemplaza todas las apariciones del string buscado con el string de reemplazo
strcmp — Comparación de string segura a nivel binario
strlen — Obtiene la longitud de una cadena

strpos — Busca la posición de la primera ocurrencia de una cadena
strrev — Invierte una string
strstr — Buscar la primera aparición de una cadena
strtr — Reemplaza caracteres dentro de una cadena
strtolower — Convierte una cadena a minúsculas
strtoupper — Convierte un string a mayúsculas

Arrays

sort – Ordena un array
rsort – Ordena en orden inverso un array

Formularios

Un ejemplo sencillo

La página inicial es ésta. Aquí solicitamos los datos. Puedes llamarla **pagina1.html**.

```
<html>
<head> <title>Formulario de entrada del datos</title> </head>
<body>
  <form method="post" action="pagina2.php">
     Nombre y Apellidos:  <input type="text" name="nombre">   <br>
     <input type="submit" value="enviar">
  </form>
</body>
</html>
```

Esta es la página que recoge los datos del formulario y los procesa, en este caso únicamente los muestra por pantalla. Su nombre es **pagina2.php**.

```
<html>
<head> <title>Captura de datos del formulario</title> </head>
<body>
<?php
  echo "El nombre ingresado es:";
  echo $_REQUEST['nombre'];
  ?>
</body>
</html>
```

Un ejemplo más completo

Este ejemplo realiza la solicitud y proceso de datos en una única página (puedes llamarla **form.php**). Aunque esto no suele ser lo normal, se hace así porque esto permite ir una y otra vez a la misma página y modificar sus datos todas las veces que se desee para pruebas. El alumno puede añadir al final un botón (algo así como "Procesar datos") en el cual saldríamos de este ciclo y dichos datos serían finalmente procesados.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />   <!-- iso-8859-1, cp1252 -->

<form action="<?php $_SERVER[PHP_SELF] ?>" method="post" >
  <div style='width: 400px; background-color: lime '>
  <b>Usuario y clave</b><br>
  <input type='text' name='nombre' value="<?php $_REQUEST[nombre] ?>"/>
  <input type='password' name='clave' value="<?php $_REQUEST[clave] ?>" />
  </div>


  <div style='width: 400px; background-color: wheat'>
  <b>Estado civil</b><br>
  <input type='radio' name='estado' value='soltero' <?php  if ($_REQUEST['estado']=="soltero") echo 'checked' ?> />Soltero <br>
  <input type='radio' name='estado' value='casado'  <?php  if ($_REQUEST['estado']=="casado") echo 'checked' ?> />Casado <br>
  </div>

 <!--
  <div style='width: 400px; background-color: wheat'>
  <b>Estado civil</b><br>
  <select name="estado">
  <option value="soltero" <?php  if ($_REQUEST['estado']=="soltero") echo 'selected' ?>>Soltero</option>
  <option value="casado" <?php  if ($_REQUEST['estado']=="casado") echo 'selected' ?>>Casado</option>
  </select>
  </div>
 -->

  <div style='width: 400px; background-color: khaki '>
  <b>Aficiones</b><br>
  <input type='checkbox' name='deporte' value='deporte' <?php  if ($_REQUEST[deporte]) echo 'checked' ?>/>Hacer deporte <br>
  <input type='checkbox' name='musica' value='musica' <?php  if ($_REQUEST[musica]) echo 'checked' ?>/>Escuchar música <br>
  <input type='checkbox' name='otros' value='otros' <?php  if ($_REQUEST[otros]) echo 'checked' ?>/>Otras aficiones <br>
  </div>

  <div style='width: 400px; background-color: tan'>
  <b>Observaciones</b><br>
  <textarea name='observaciones' cols='40' rows='5'><?php echo "$_REQUEST[observaciones]" ?></textarea> <br>
  </div>

  <div style='width: 400px; background-color: lightsalmon'>
  <b>Foto</b><br>
  <input type="file" name="foto"><br><br>
  </div>

  <input type="hidden" name="oculto" value="Algo oculto">
```

```
  <input type="button" value="Imprimir formulario" onclick='window.print()'>
  <input type="reset" value="Resetear" />
  <input type="button" value="Borrar todo" onclick="location='<?php echo "http://$_SERVER[SERVER_NAME]$_SERVER[PHP_SELF]" ?>'" />
  <input type='submit' value='Enviar' />
  <input type="image" name="enviar" src="enviar.png" /> <!-- Para submit -->
</form>



<?php

if ($_REQUEST) {
    // Mostramos valores del formulario
    echo '<pre>';
    print_r($_REQUEST);
    echo '</pre>';
}

?>
```

Subida de archivos (Upload)

Una actividad común en un sitio es poder almacenar un archivo en el servidor, más comúnmente conocido como upload.
La primera página la llamaremos **foto1.html** y contiene el formulario para solicitar el archivo:

```
<html>
<head> <title>Foto a subir</title> </head>
<body>
    <form action="foto2.php" method="post" enctype="multipart/form-data">
    Seleccione el archivo: <input type="file" name="foto"><br>
    <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

Veamos los puntos que tenemos que respetar cuando efectuamos el upload de archivos:

```
<form action="foto2.php" method="post" enctype="multipart/form-data">
```

Se define un nuevo atributo para el formulario (**enctype="multipart/form-data**), con esta indicamos que dentro del formulario de carga se envían archivos. Hay que tener mucho cuidado cuando escribimos este atributo, si nos equivocamos en algún carácter el proceso de upload no funcionará.

El control HTML para la selección del archivo es de tipo "file":

```
<input type="file" name="foto">
```

Automáticamente aparecerá el botón dentro de la página para poder navegar en nuestro disco duro para la selección del archivo (por defecto PHP está configurado para poder cargar archivos de hasta 2 Mb, de todos modos, a este valor lo podemos modificar).

La página (**foto2.php**) que guarda el archivo en el servidor es:

```
<html>
<head> <title>Guardar foto <?php echo $_FILES["foto"]["name"] ?></title> </head>
<body>
  <?php if (move_uploaded_file($_FILES["foto"]["tmp_name"], $_FILES["foto"]["name"]))  {?>
      <b>Foto subida</b><br>
      <img src="<?php echo $_FILES["foto"]["name"] ?>"><br>
      Nombre: <?php echo $_FILES["foto"]["name"] ?><br>
      Tipo: <?php echo $_FILES["foto"]["type"] ?><br>
      Bytes: <?php echo $_FILES["foto"]["size"] ?>
  <?php } else { ?>
      <b>Hubo un error al subir la foto.</b>
  <?php } ?>
</body>
</html>
```

Cuando se ejecuta esta página, ya está almacenado en el servidor el archivo, en una carpeta temporal. Ahora nos hace falta mover el mismo a la carpeta donde se encuentra nuestra página (en definitiva nuestro sitio de internet). Para moverla desde el directorio temporal hacemos:

```
move_uploaded_file($_FILES["foto"]["tmp_name"], $_FILES["foto"]["name"])
```

> ***NOTA**: Aunque nosotros hemos subido una imagen, con esta técnica se puede subir cualquier tipo de archivo.*

Cookies y Sesiones

El protocolo HTTP es un protocolo sin estado. Esto quiere decir cada vez que solicitamos una página a un servidor representa una conexión distinta. Una cookie es una pequeña cantidad de datos que el servidor envía para que la almacene el cliente. Una cookie está limitada a 4KB.

Luego que una cookie es creada sólo el servidor que la creó puede leerla. Cada vez que el navegador del usuario visita el sitio, se envía dicha cookie. Otra cosa importante que hay que tener en cuenta es que el usuario del navegador puede configurar el mismo para no permitir la creación de cookies, lo que significa que el uso de cookies debe hacerse con moderación y cuando la situación lo requiera. De todos modos, el 95% de los navegadores están configurados para permitir la creación de cookies.

Una cookie consta de un nombre, un valor, una fecha de expiración y un servidor.
Para la creación de una cookie desde PHP debemos llamar a la función setcookie:
setcookie(<nombre de la cookie>, <valor de la cookie>, <fecha de expiración>, <carpeta del servidor>)

Con un problema sencillo entenderemos el uso de esta función. Supongamos que queremos que los usuarios que entran a nuestro sitio puedan configurar con qué color de fondo de página quiere que aparezca cada vez que ingresa al sitio. Al color seleccionado por el visitante lo almacenaremos en una cookie. En caso que no exista el color, por defecto es blanco.

La primera página mostrará un formulario con tres controles de tipo radio para la selección del color. También esta página verificará si existe la cookie creada, en caso afirmativo fijará el fondo de la página con el valor de la cookie. Tengamos en cuenta que la primera vez que ejecutemos este programa la página es de color blanco, luego variará según el color seleccionado en el formulario.
El código de la primera página es:

```
<html>
<head> <title>Leemos la cookie si existe</title> </head>
<body <?php if (isset($_COOKIE['color'])) echo " bgcolor=\"$_COOKIE[color]\"" ?>>
    <form action="cookie2.php" method="post">
    Seleccione de que color desea que sea la página de ahora en más:<br>
    <input type="radio" value="rojo" name="radio">Rojo<br>
    <input type="radio" value="verde" name="radio">Verde<br>
    <input type="radio" value="azul" name="radio">Azul<br>
    <input type="submit" value="Crear cookie">
    </form>
</body>
</html>
```

El formulario no varía en nada respecto a otros vistos. Lo más importante es el bloque PHP que verifica si ya existe la cookie en el navegador del cliente. Es importante entender que la primera vez que ejecutemos esta página la cookie no existe, por lo que el if se verifica falso:

```
<body <?php if (isset($_COOKIE['color'])) echo " bgcolor=\"$_COOKIE[color]\"" ?> >
```

El vector asociativo $_COOKIE almacena todas las cookies creadas por el visitante. Si es la primera vez que peticionamos esta página, el vector $_COOKIE no tendrá elementos. Es decir que la marca body no tiene inicializada la propiedad bgcolor.

La segunda página es la que crea la cookie propiamente dicha:

```
<?php
if ($_REQUEST['radio']=="rojo")
  setcookie("color","#ff0000",time()+60*60*24*365,"/");
elseif ($_REQUEST['radio']=="verde")
  setcookie("color","#00ff00",time()+60*60*24*365,"/");
elseif ($_REQUEST['radio']=="azul")
  setcookie("color","#0000ff",time()+60*60*24*365,"/");
?>
<html>
<head> <title>Creamos la cookie</title> </head>
<body>
  Se creó la cookie. <br>
  <a href="cookie1.php">Ir a la página anterior</a>
</body>
</html>
```

La llamada a la función setcookie debe hacerse antes de imprimir cualquier marca HTML, de lo contrario no funcionará.
Como podemos observar, la creación de la cookie se hace llamando a la función setcookie:

```
  setcookie("color", "#ff0000", time()+60*60*24*365, "/");
```

El nombre de la cookie se llama "color" y el valor que almacenamos depende de qué control de tipo radio esté seleccionado en la página anterior. La fecha de expiración de la cookie la calculamos fácilmente llamando a la función time() que nos retorna la fecha actual en segundos y le sumamos el producto 60*60*24*365 (60 segundos * 60 minutos * 24 horas * 365 días) es decir que la cookie existirá en la máquina del visitante hasta el año próximo. Cuando indicamos como directorio la sintaxis "/" significa que la cookie se crea a nivel del sitio y con cuaquier petición a dicho sitio, el navegador enviará la cookie al servidor.
Por último dispusimos en esta página un hipervínculo a la página anterior, para ver que, de ahora en más, cada vez que ejecutemos la pagina1.php, el color de fondo de la misma dependerá del valor de la cookie registrada.

Eliminar una cookie

Para borrar una cookie se debe llamar a la función setcookie con una fecha anterior a la actual. Por ejemplo para eliminar la cookie anterior hacemos:

```
setcookie("color", "#ff0000", time()-1000, "/");
```

> ***NOTA**: Recuerda que la función **setcookie** debe utilizarse antes de cualquier etiqueta html, al principio del script.*

Cookies de sesión

Una cookie de sesión es aquella que sólo existe mientras no cerremos el navegador. Una vez cerrado el navegador la cookie desaparece. Para crear este tipo de cookies se pone el atributo de tiempo a 0. Por ejemplo para la cookie anterior sería:

```
setcookie("color", "#ff0000", 0, "/");
```

> ***NOTA**: Recuerda que la función **setcookie** debe utilizarse antes de cualquier etiqueta html, al principio del script.*

Variables de sesión

Es otro método para hacer que variables estén disponibles en múltiples páginas sin tener que pasarlas como parámetro. A diferencia de las cookies, las variables de sesión se almacenan en el servidor y tienen un tiempo limitado de existencia.
Para identificar al usuario que generó las variables de sesión, el servidor genera una clave única que es enviada al navegador y almacenada en una cookie. Luego, cada vez que el navegador solicita otra página al mismo sitio, envía esta cookie (clave única) con la cual el servidor identifica de qué navegador proviene la petición y puede rescatar de un archivo de texto las variables de sesión que se han creado. Cuando han pasado 20 minutos sin peticiones por parte de un cliente (navegador) las variables de sesión son eliminadas automáticamente (se puede configurar el entorno de PHP para variar este tiempo).

Una variable de sesión es más segura que una cookie ya que se almacena en el servidor. Otra ventaja es que no tiene que estar enviándose continuamente como sucede con las cookies. Otra ventaja de emplear una variable de sesión en lugar de una cookie es que cuando el navegador del cliente está configurado para desactivar las cookies las variables de sesión, tienen forma de funcionar (enviando la clave como parámetro en cada hipervínculo).

Como desventaja podemos decir que ocupa espacio en el servidor.

Haremos un problema muy sencillo, cargaremos en un formulario el nombre de usuario y clave de un cliente, en la segunda página crearemos dos variables de sesión y en una tercera página recuperaremos los valores almacenados en las variables de sesión.
La primera página es un formulario HTML puro:

```
<html>
<head> <title>Pagina 1</title> </head>
<body>
    <form action="pagina2.php" method="post">
    Ingrese nombre de usuario:
    <input type="text" name="campousuario"><br>
    Ingrese clave:
    <input type="password" name="campoclave"><br>
    <input type="submit" value="confirmar">
    </form>
</body>
</html>
```

La segunda página es donde creamos e inicializamos las dos variables de sesión:

```
<?php
session_start();
$_SESSION['usuario']=$_REQUEST['campousuario'];
$_SESSION['clave']=$_REQUEST['campoclave'];
?>
<html>
<head> <title>Pagina 2</title></head>
<body>
    Se almacenaron dos variables de sesión.<br><br>
    <a href="pagina3.php">Ir a la tercer página donde se recuperarán las variables de sesión</a>
</body>
</html>
```

> ***NOTA**: Recuerda que la función **session_start** debe utilizarse antes de cualquier etiqueta html, al principio del script.*

Cuando creamos o accedemos al contenido de variables de sesión debemos llamar a la función **session_start()** antes de cualquier salida de marcas HTML. Para almacenar los valores en las variables de sesión lo hacemos:

```
$_SESSION['usuario']=$_REQUEST['campousuario'];
$_SESSION['clave']=$_REQUEST['campoclave'];
```

Es decir, tenemos el vector asociativo $_SESSION que almacena las variables de sesión.
Por último, esta página tiene un hipervínculo a la tercera página.

La última página de este ejemplo tiene por objetivo acceder a las variables de sesión:

```
<?php
session_start();
?>
<html>
<head> <title>Pagina 3</title> </head>
<body>
<?php
echo "Nombre de usuario recuperado de la variable de sesión:".$_SESSION['usuario'];
echo "<br><br>";
echo "La clave recuperada de la variable de sesión:".$_SESSION['clave'];
?>
</body>
</html>
```

De nuevo vemos que la primera línea de esta página es la llamada a la función **session_start()** que, entre otras cosas, rescata de un archivo de texto las variables de sesión creadas para ese usuario (recordemos que desde el navegador todas las veces retorna una cookie con la clave que generó PHP la primera vez que llamamos a una página del sitio). Para mostrar las variables de sesión, las accedemos por medio del vector asociativo $_SESSION:

```
echo "Nombre de usuario recuperado de la variable de sesión:".$_SESSION['usuario'];
echo "<br><br>";
echo "La clave recuperada de la variable de sesión:".$_SESSION['clave'];
```

> ***NOTA**: Tengamos en cuenta que en cualquier otra página del sitio tenemos acceso a las variables de sesión sólo con llamar inicialmente a la función session_start().*

Archivos

Funciones para archivos y directorios

getcwd — Obtiene el directorio actual en donde se esta trabajando
chdir — Cambia de directorio
scandir — Lista los archivos y directorios ubicados en la ruta especificada
mkdir — Crea un directorio
rmdir — Elimina un directorio vacío

finfo_file – Información acerca de un archivo
touch — Cambia fecha o crea archivo vacío
copy — Copia archivos
rename — Renombra un archivo o directorio
delete / unlink — Borra un archivo

pathinfo — Devuelve información acerca de una ruta de archivo
basename — Devuelve el componente de nombre de rastreo de la ruta
dirname — Devuelve el directorio padre de la ruta
realpath — Devuelve el nombre de la ruta absoluta canonizada

filesize — Obtiene el tamaño de un archivo
filetype — Obtiene el tipo de archivo

is_dir — Indica si el nombre de archivo es un directorio
is_executable — Indica si el nombre de archivo es ejecutable
is_file — Indica si el nombre de archivo es un archivo normal
is_link — Indica si el nombre de archivo es un enlace simbólico
is_readable — Indica si un archivo existe y es legible
is_writable / is_writeable — Indica si un archivo existe y es escribible

fopen — Abre un archivo o URL
fclose — Cierra un puntero a un archivo abierto
feof — Comprueba si el puntero a un archivo está al final del archivo

fputs — Alias de fwrite
fread — Lectura de un fichero en modo binario seguro
fwrite — Escritura de un archivo en modo binario seguro
fseek — Busca sobre un puntero a un archivo
ftell — Devuelve la posición de lectura/escritura actual del puntero a un archivo

fflush — Vuelca la salida a un archivo
fgetc — Obtiene un carácter de un puntero a un archivo
fgets — Obtiene un línea del puntero a un archivo

file_exists — Comprueba si existe un archivo o directorio
file_get_contents — Transmite un archivo entero a una cadena
file_put_contents — Escribe una cadena a un archivo
readfile — Transmite un archivo
file — Transfiere un archivo completo a una matriz

Como habrás apreciado más arriba, existen numerosas funciones para el trabajo con carpetas y archivos. Existen algunas más pero las más importantes son las que se muestran. El núcleo "duro" de estas funciones son fopen, fclose, feof, fread, fgets, fwrite, fputs.
fopen abre un archivo o URL. Se le pasa como segundo parámetro el modo de apertura.

```
<?php
  $archivo = fopen("/home/rasmus/archivo.txt", "r");
  $archivo = fopen("/home/rasmus/archivo.gif", "wb");
  $archivo = fopen("http://www.example.com/", "r");
  $archivo = fopen("ftp://user:password@example.com/archivo.txt", "w");
?>
```

Los modos más frecuentes son:

modo	Descripción
'r'	Apertura para sólo lectura; coloca el puntero al archivo al principio del archivo.
'r+'	Apertura para lectura y escritura; coloca el puntero al archivo al principio del archivo.
'w'	Apertura para sólo escritura; coloca el puntero al archivo al principio del archivo y trunca el archivo a longitud cero. Si el archivo no existe se intenta crear.
'w+'	Apertura para lectura y escritura; coloca el puntero al archivo al principio del archivo y trunca el archivo a longitud cero. Si el archivo no existe se intenta crear.
'a'	Apertura para sólo escritura; coloca el puntero al archivo al final del archivo. Si el archivo no existe se intenta crear.
'a+'	Apertura para lectura y escritura; coloca el puntero al archivo al final del archivo. Si el archivo no existe se intenta crear.

Las funciones fgets y fread permiten leer desde un archivo, mientras que las funciones fputs y fwrite permiten escribir en un archivo. Se utilizan las funciones fgets y fputs para archivos tipo texto y fread y fwrite para archivos binarios.

Ejemplo de fread

```php
<?php
$nombre_archivo = "c:\\files\\imagen.gif";
$archivo = fopen($nombre_archivo, "rb");
$contenido = fread($archivo, filesize($nombre_archivo));
fclose($archivo);
?>
```

Ejemplo de fwrite

```php
<?php
$nombre_archivo = 'prueba.txt';
$contenido = "Añade esto al archivo\n";

if (is_writable($nombre_archivo)) {
    $archivo = fopen($nombre_archivo, 'a'));
    fwrite($archivo, $contenido));
    fclose($archivo);
} else {
    echo "El archivo $nombre_archivo no es escribible";
}
?>
```

Ejemplo de fgets

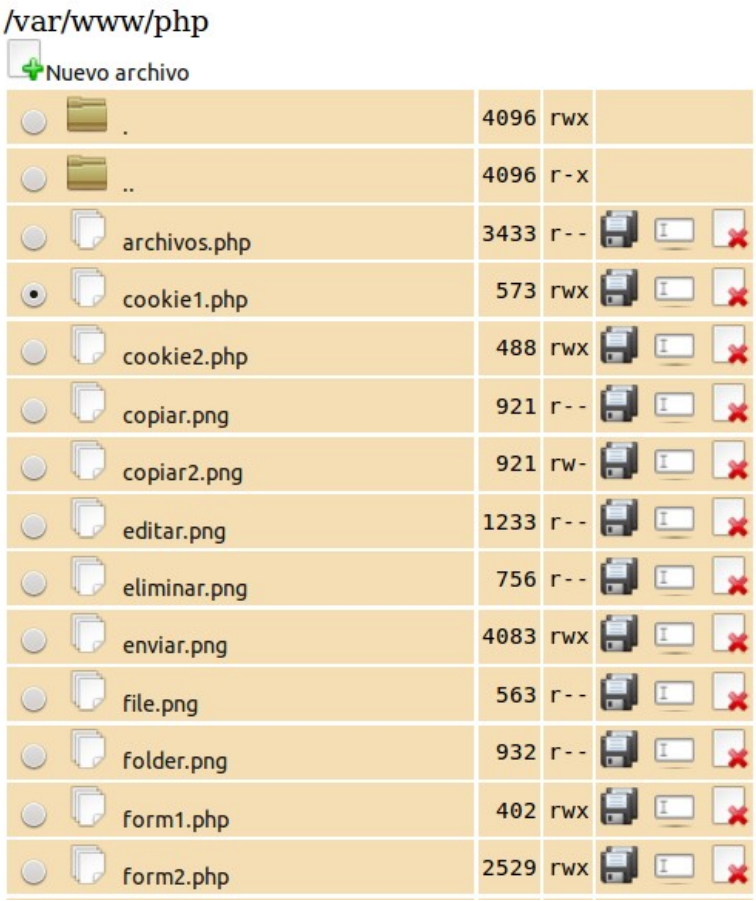
```
<html>
<head><title>Lectura en un archivo</title></head><body>
<?php
  $ar=fopen("datos.txt","r") or die("No se pudo abrir el archivo");
  while (!feof($ar))
  {
    $linea=fgets($ar);
    echo "$linea<br>";
  }
  fclose($ar);
  ?>
</body>
</html>
```

Ejemplo de fputs

```
<html>
<head><title>Escritura en un archivo</title></head>
<body>
<?php
  $ar=fopen("datos.txt","a") or die("Problemas en la creacion");
  fputs($ar,"Prueba de escritura en un archivo");
  fputs($ar,"\n");
  fputs($ar,"----------------------------------------------------------");
  fputs($ar,"\n");
  fclose($ar);
  ?>
</body>
</html>
```

Las funciones file_get_contents y file_put_contents permiten abrir, leer o escribir y cerrar un archivo, todo ello en una única sentencia. Las funciones touch, copy, rename y unlink permiten crear nuevo, copiar, renombrar y borrar un archivo respectivamente. Con file_info obtenemos información acerca de un archivo. Estas y algunas funciones las vamos a ver con un ejemplo. Lo siguiente es un explorador de archivos realizado en PHP. Primero tienes la imagen y luego el código (explorador.php).

Ejemplo de explorador de archivos



cookie1.php - 573 bytes

```
<html>
<head> <title>Leemos la cookie si existe</title>
    <script type="text/javascript">
    </script>
</head>
<body <?php if (isset($_COOKIE['color'])) echo "
bgcolor=\"$_COOKIE[color]\"" ?>>
    <form action="cookie2.php" method="post">
    Seleccione de que color desea que sea la página de ahora en
más:<br>
    <input type="radio" value="rojo" name="radio">Rojo<br>
    <input type="radio" value="verde" name="radio">Verde<br>
    <input type="radio" value="azul" name="radio">Azul<br>
    <input type="submit" value="Crear cookie">
    </form>
</body>
</html>
```

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<!-- Explorador de archivos y carpetas -- JAMJ Octubre-Noviembre 2011 -->
<!-- Es necesaria la extensión fileinfo de PHP -->
<style type="text/css">
    #izquierda     { float: left; min-width: 50%; }
    #derecha       { position: absolute; right: 0;  min-width: 49%; }
    textarea       { width: 100%; min-height: 400px; }
    .linea, .bytes { background-color: wheat; border: solid wheat 1px; font-family: monospace;}
    .linea:hover   { background-color: tan  ; border: solid chocolate 1px; }
    .bytes         { text-align: right;}
    input          { background-color: inherit; border: background hidden; }
    input:hover    { background-color: inherit; border: background hidden; cursor: pointer;}
</style>
</head>

</body>

<?php
// Procesamos al inicio las operaciones de Nuevo, Copiar, Renombar y Eliminar
if ($_REQUEST[editar_x])    // Editar archivo
    file_put_contents ("$_REQUEST[dir_actual]/$_REQUEST[nombre]", "$_REQUEST[datos]");

if ($_REQUEST[nuevo_x])     // Nuevo archivo
    touch ("$_REQUEST[dir_actual]/$_REQUEST[crear_nuevo]");

if ($_REQUEST[eliminar_x])  // Eliminar archivo
    unlink ("$_REQUEST[dir_actual]/$_REQUEST[nombre]");

if ($_REQUEST[renombrar_x]) // Renombar archivo
    rename ("$_REQUEST[dir_actual]/$_REQUEST[nombre]", $_REQUEST[dir_actual]/$_REQUEST[strtr($_REQUEST[nombre], ". ", "__")] );

if ($_REQUEST[copiar_x])    // Copiar archivo
    copy ("$_REQUEST[dir_actual]/$_REQUEST[nombre]", "$_REQUEST[dir_actual]/$_REQUEST[nombre] (copia)" );

if (isset($_REQUEST[dir_actual])) {
  if (is_dir("$_REQUEST[dir_actual]/$_REQUEST[nombre]"))
    chdir ("$_REQUEST[dir_actual]/$_REQUEST[nombre]");
  else
    chdir ("$_REQUEST[dir_actual]");
}

$dir_actual=getcwd();
```

```
?>

<form action="<?php echo $_SERVER[PHP_SELF] ?>" method="post" >
<div id="izquierda">

<?php echo "$dir_actual<br>"; ?>
<table>
<input type='image' name='nuevo' id='nuevo' src='nuevo.png'>
<input id='crear_nuevo' name='crear_nuevo' type='text' value='Nuevo archivo' onfocus='clear()'>

<?php

$archivos=scandir(getcwd());
foreach ($archivos as $a) {
?>
<tr>
    <td class="linea">
    <input type="radio" name="nombre" value="<?php echo $a ?>"
          <?php if ($_REQUEST[nombre]==$a) echo "checked" ?> onclick='submit()'>
    <img src="<?php if (is_dir($a)) echo 'folder.png'; else echo 'file.png'; ?>">

    <input type="text" name="<?php echo $a ?>" value="<?php echo $a ?>">
    </td>
    <td class="linea bytes">
        <?php echo filesize($a); ?>
    </td>
    <td class="linea">
        <?php
        if (is_readable($a))    echo "r"; else echo "-";
        if (is_writable($a))    echo "w"; else echo "-";
        if (is_executable($a))  echo "x"; else echo "-";
        ?>
    </td>
    <td class="linea">
        <?php if (!is_dir($a)) { ?>
        <input type="image" src="copiar.png"  name="copiar" />
        <input type="image" src="renombrar.png" name="renombrar" />
        <input type="image" src="eliminar.png" name="eliminar" >
        <?php } ?>
    </td>
</tr>

<?php } ?>
```

```
    <input type="hidden" name="dir_actual" value="<?php echo $dir_actual ?>">
</table>
</div>


<div id="derecha">
<?php
$tipo=finfo_file(finfo_open(FILEINFO_MIME_TYPE), "$_REQUEST[dir_actual]/$_REQUEST[nombre]");
if (strstr($tipo, "text") || strstr($tipo, "empty")) {
    echo "<input type='image' src='editar.png'  name='editar' />";
    echo "$_REQUEST[nombre] - ".filesize($_REQUEST[nombre])." bytes  <br>";
    $datos=file_get_contents ("$_REQUEST[dir_actual]/$_REQUEST[nombre]");
    echo "<textarea name='datos'>$datos</textarea>";
}
?>
</div>
</form>

</body>
</html>
```

Extensiones de PHP (MySQL)

Tanto para este apartado como para el siguiente necesitamos que PHP posea las extensiones adecuadas. Para este caso debe de estar instalada la extensión para MySQL. Esta extensión permite comunicarnos con un servidor de BB.DD. MySQL. Podemos ver las extensiones cargadas con:

```
<?php
        echo "<pre>";
        print_r (get_loaded_extensions());
        // phpinfo ();
        echo "</pre>";
?>
```

Funciones más útiles

mysql_client_encoding — Devuelve el nombre de la colección de caracteres
mysql_set_charset — Establece el conjunto de caracteres del cliente

mysql_connect — Abre una conexión al servidor MySQL
mysql_close — Cierra la conexión de MySQL
mysql_select_db — Seleccionar una base de datos MySQL
mysql_query — Enviar una consulta MySQL
mysql_fetch_array — Recupera una fila de resultado como un array asociativo, un array numérico o como ambos
mysql_error — Devuelve el texto con error del mensaje de la anterior operación MySQL
mysql_free_result — Libera la memoria del resultado

mysql_create_db — Crea una base de datos MySQL
mysql_drop_db — Omite (elimina) una base de datos MySQL

mysql_list_dbs — Lista de las bases de datos disponibles en un servidor MySQL
mysql_db_name — Recupera nombre de la base de datos de la llamada a mysql_list_dbs

mysql_num_rows — Obtener el número de filas de un resultset
mysql_num_fields — Obtiene el número de campos en un resultado

Listado de bases de datos disponibles

```php
<?php
error_reporting(E_ALL);

$link = mysql_connect('dbhost', 'username', 'password');
$db_list = mysql_list_dbs($link);

$i = 0;
```

```
$cnt = mysql_num_rows($db_list);
while ($i < $cnt) {
    echo mysql_db_name($db_list, $i) . "\n";
    $i++;
}
?>
```

Ejemplo general de la extensión MySQL

```
<?php
// Conectando, seleccionando la base de datos
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password') or die('No se pudo conectar: ' . mysql_error());
echo 'Conexión con éxtio';
mysql_select_db('my_database') or die('No se pudo seleccionar la base de datos');

// Realizar una consulta MySQL
$query = 'SELECT * FROM my_table';
$result = mysql_query($query) or die('Consulta fallida: ' . mysql_error());

// Imprimir los resultados en HTML
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";

// Liberar resultados
mysql_free_result($result);

// Cerrar la conexión
mysql_close($link);
?>
```

Extensiones de PHP (GD)

La extensión GD permite crear imágenes de forma dinámica. Necesitamos tener configurado PHP con esta extensión.

Funciones más frecuentes

gd_info — Reúne información acerca de la biblioteca GD instalada actualmente
imageantialias — Permite o no el uso de funciones antialias

imagegrabscreen — Capturar la pantalla completa
imagegrabwindow — Capturar una ventana

imagecreate — Crea una nueva imagen basada en paleta
imagecreatefromgif — Crea una nueva imagen a partir de un fichero o de una URL
imagecreatefromjpeg — Crea una nueva imagen a partir de un fichero o de una URL
imagecreatefrompng — Crea una nueva imagen a partir de un fichero o de una URL
imagecreatefromstring — Crear una imagen nueva desde el flujo de imagen de la cadena
imagecreatetruecolor — Crear una nueva imagen de color verdadero
imagedestroy — Destruir una imagen
imagefilter — Aplica un filtro a una imagen
imagecopy — Copiar parte de una imagen
imagecopymerge — Copiar y fusionar parte de una imagen
imagecopymergegray — Copiar y fusionar parte de una imagen con escala de grises
imagecopyresampled — Copia y cambia el tamaño de parte de una imagen redimensionándola
imagecopyresized — Copia y cambia el tamaño de parte de una imagen
imagerotate — Rotar una imagen con un ángulo dado
imagesetbrush — Establecer la imagen de pincel para el dibujo de líneas
imagesetpixel — Establecer un simple píxel
imagesetstyle — Establecer el estilo para el dibujo de líneas
imagesetthickness — Establecer el grosor para el dibujo de líneas

imagejpeg — Exportar la imagen al navegador o a un fichero
imagegif — Exportar la imagen al navegador o a un fichero
imagepng — Imprimir una imagen PNG al navegador o a un archivo

imagewbmp — Exportar la imagen al navegador o a un fichero
imagexbm — Imprimir una imagen XBM al navegador o a un archivo

imagecolorallocate — Asigna un color para una imagen
imagecolorallocatealpha — Asigna un color para una imagen
imagecolordeallocate — Desasignar un color de una imagen

imageline — Dibujar una línea
imagearc — Dibujar un arco
imagerectangle — Dibuja un rectángulo
imagepolygon — Dibujar un polígono
imageellipse — Dibujar una elipse
imagefill — Rellenar
imagefilledarc — Dibujar un arco parcial y rellenarlo
imagefilledrectangle — Dibujar un rectángulo con relleno
imagefilledpolygon — Dibujar un polígono con relleno
imagefilledellipse — Dibujar una elipse con relleno

imagestring — Dibujar una cadena horizontalmente
imagestringup — Dibujar una cadena verticalmente
imageloadfont — Cargar una nueva fuente
imagettftext — Escribir texto en la imagen usando fuentes TrueType
imagefttext — Escribir texto en la imagen usando fuentes mediante FreeType 2

Ejemplo de como utilizar fuente TTF

Mostramos “Hola mundo” en una imagen haciendo uso de fuente TTF.

```php
<?php
// Crear una imagen de 300x100
$im = imagecreatetruecolor(300, 100);
$rojo = imagecolorallocate($im, 0xFF, 0x00, 0x00);
$negro = imagecolorallocate($im, 0x00, 0x00, 0x00);

// Hacer el fondo rojo
imagefilledrectangle($im, 0, 0, 299, 99, $rojo);

// Ruta a nuestro archivo de fuente ttf
$archivo_fuente = './fuente.ttf';

// Dibuja el texto 'Hola mundo' usando un tamaño de fuente de 13 y rotación 0 grados
// en la posición x,y = 105,55
imagefttext($im, 13, 0, 105, 55, $negro, $archivo_fuente, 'Hola mundo');

// Imprimir la imagen al navegador
header('Content-Type: image/png');
imagepng($im);
imagedestroy($im);
?>
```

Ejemplo de acceso restringido mediante captcha

El siguiente ejemplo y explicaciones han sido adaptados tomando como base los disponibles en http://www.phpya.com.ar .

A menudo cuando navegamos por Internet y rellenamos algún formulario (para un alta en el correo electrónico o similar) nos aparece una imagen con un texto o un número que debemos introducir para evitar altas automáticas **captcha**). En este ejemplo vamos a implementar dicha funcionalidad.

Mostraremos al usuario un formulario (**pagina1.html**) y deberá introducir el número generado aleatoriamente por el guión **pagina2.php**. En la tercera página (**pagina3.php**) finalizamos el proceso y ofrecemos o denegamos el servicio.

Veamos primero como generamos una imagen con un número aleatorio y dificultamos su visibilidad para evitar el reconocimiento OCR.

El segundo archivo "**pagina2.php**" es la imagen propiamente dicha:

```php
<?php
$ancho=100;
$alto=30;
$imagen=imageCreate($ancho,$alto);

// Generamos fondo amarillo
$amarillo=ImageColorAllocate($imagen,255,255,0);
ImageFill($imagen,0,0,$amarillo);

// Generamos número aleatorio y guardamos en variable de sesión
$valoraleatorio=rand(100000,999999);
session_start();
$_SESSION['numeroaleatorio']=$valoraleatorio;

// Dibujamos número con color rojo
$rojo=ImageColorAllocate($imagen,255,0,0);
ImageString($imagen,5,25,5,$valoraleatorio,$rojo);

// Generamos 6 líneas para dificultar lectura a programas OCR
for($c=0;$c<=5;$c++)
{
  $x1=rand(0,$ancho);
  $y1=rand(0,$alto);
  $x2=rand(0,$ancho);
  $y2=rand(0,$alto);
  ImageLine($imagen,$x1,$y1,$x2,$y2,$rojo);
}
// Enviamos imagen al navegador
Header ("Content-type: image/jpeg");
ImageJPEG ($imagen);
ImageDestroy($imagen);
?>
```

En este guión podemos resaltar dos cosas:

- Guardamos el número aleatorio en una variable de sesión para poder recuperarla en pagina3.php:

  ```php
  session_start();
  $_SESSION['numeroaleatorio']=$valoraleatorio;
  ```

- La función ImageString imprime el número generado de color rojo en las coordenadas 25,5 y con un tamaño de fuente 5 (valores posibles de fuente son de 1 a 5).

Para mostrar el formulario y la imagen dinámica antes creada escribimos **pagina1.html**:

```
<html>
<head><title>Captcha</title></head>
<body>
    <form action="pagina3.php" method="post">
    Dígitos verificadores:<img src="pagina2.php"><br>
    Ingrese valor: <input type="text" name="numero"><br>
    <input type="submit" value="Verificar">
    </form>
</body>
</html>
```

Tengamos en cuenta que ésta es la página que se solicita inicialmente. Dentro de esta página, se incorpora una marca img para agregar una imagen a la página, pero la misma no es un archivo estático sino un archivo PHP que genera la imagen:

```
Dígitos verificadores:<img src="pagina2.php"><br>
```

Como se indica, la página que procesa el valor ingresado y el valor generado en forma aleatoria es la tercera página:

```
<form action="pagina3.php" method="post">
```

Por último el tercer archivo "**pagina3.php**":

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

<html>
<head> <title>Finalizamos</title> </head>
<body>
<?php
if ($_SESSION['numeroaleatorio']==$_REQUEST['numero'])
  echo "Todo correcto";
else
  echo "Incorrecto";
?>
</body>
</html>
```

Lo primero que hacemos es llamar a la función que rescata las variables de sesión:

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

Después comprabamos el valor que se nos pasa mediante $_REQUEST con el valor de $_SESSION y en función de ello comprobamos si se introdujo correctamente o no.

```
if ($_SESSION['numeroaleatorio']==$_REQUEST['numero'])
```

En este caso únicamente mostramos si todo ha ido correcto o no, pero podemos modificar la página para finalizar el registro u ofrecer un servicio en caso de que el usuario haya realizado todos los pasos correctamente.