Constantes

Definición

Además de las variables (que pueden ser reasignadas), PHP ofrece constantes, que tienen un único valor en toda su vida. Las constantes no tienen un $ delante de sus nombres y por convenio los nombres de las mismas por lo general están en letras mayúsculas. Las constantes sólo pueden contener valores escalares (números y cadena).

Las constantes tienen alcance mundial para que sean accesibles en todas partes en las secuencias de comandos después de que han sido definidos, incluso dentro de las funciones.

```
define(IVA, 21);
```

IVA causaría para evaluar a 21 en todas las partes en las que aparece su código. No hay
forma de cambiar esta tarea después de que se ha hecho, en cambio sí se podría en caso de de variables.

Características

- Es un identificador que expresa un valor fijo.
- Su valor no puede ser modificado ni eliminado una vez que se ha definido.
- Las constantes son como las variables con la diferencia que no pueden ser redefinidas.
- Otra diferencia con las variables es que las constantes no llevan el prefijo “$”.
- Las constantes son sensibles de las mayúsculas y minúsculas.
- Las constantes sólo pueden ser definidas a través de la función define()
- Por convención, en PHP los nombres de constantes suelen identificarse con mayúsculas únicamente. Esto facilita una rápida lectura de un código PHP y la fácil diferenciación con las variables.
- Las constantes no pueden almacenar arreglos de caracteres, sólo valores escalares.
- Las constantes se definen de la siguiente forma: define(“nombre_constante”, valor);

Ejemplo de uso de Constantes

```
<?

define(“PI”, 3.14);

echo PI;        // muestra el valor 3.14

?>
```

Ejemplo Completo de uso de Constantes

```
1. <?php
2. $titulo = "Uso de Constantes" ;
3. $encabezado = "Ejemplo de Uso de Constantes" ;
4. ?>

5. <html>
6. <head>
7. <title> <?php echo $titulo; ?> </title>
8. </head>
9. <body>

10. <h1 align="center"> <? echo $encabezado; ?>  </h1>
11. <BR>

12. <form name="frm" method=post action="eje7.php">

13. Ingrese un valor: <input name=valor size=20>

14. <input type=submit value=Enviar>

15. </form>

16. <?php
17. define ("PORC",15);

18. echo "El monto ingresado fue: " . $valor . "<br>\n" ;
19. echo "El porcentaje es: " . PORC . "% <br>\n" ;

20. $descuento = $valor * PORC /100 ;
```

```
21. echo "El descuento es: " . $descuento . "<br>\n" ;
22. ?>

23. </body>
24. </html>
```

Predefinidas

Son valores constantes ya predefinidos por PHP.

Son palabras reservadas del lenguaje.

Algunas constantes nativas de PHP

TRUE	**Valor Verdadero**
FALSE	**Valor Falso**
PHP_OS	**Sistema Operativo donde se ejecuta el analizador de PHP**
PHP_VERSION	**Versión del analizador de PHP**
M_PI	**El Valor ∏ (3.1415926535898)**
DIRECTORY_SEPARATOR	**El símbolo de separación de directorios "\"**

Manejo de Archivos

Introducción

Aquí comienza el principio de persistencia de datos. Con persistencia referimos a la posibilidad de almacenar información (en el Server) y que los datos persistan en el tiempo para poder acceder a los mismos en otro momento cuando sea necesario.

Existen dos formas de almacenar datos. Una es mediante un archivo de texto, la otra es mediante la utilización de bases de datos.

Muchas veces se necesita almacenar información para su uso posterior, sin la necesidad de acceder a una base de datos como podría ser mySQL, SQL Server, Oracle, etc. Esto sería información ágil y generalmente breve, por lo cual no vale la pena la utilización de bases de datos para cosas simples. El manejo de los archivos de texto se convierte en una prioridad cuando se desarrollan sitios web y se requiere de que se cree, guarde, lea, escriba o borre archivos. Algunos ejemplos para los que se utilizan archivos de texto en PHP son: un contador de visitas, un archivo de log de errores, un archivo de configuración del sirio, etc.

¿Cómo se leen los datos en un archivo de texto?

- Abrir el archivo (si no existe se avisa con un mensaje).
- Leer los datos.
- Cerrar el archivo.

¿Cómo se graban los datos en un archivo de texto?

- Abrir el archivo (si no existe se lo creará).
- Grabar los datos en el archivo.
- Cerrar el archivo.

Abrir y cerrar archivos de texto

Función fopen()

La función fopen() se utiliza para abrir los archivos de texto. Su sintaxis es:

fopen(nombre del archivo, modo)

El primer parámetro es el nombre del archivo, mientras que el segundo es el modo en que se utilizará el archivo. La siguiente es una lista de los diferentes modos para operar con un archivo de texto:

r	Abre el archivo en modo de lectura, empezando por la parte inicial del archivo. O sea, que el puntero del archivo se posiciona por defecto en el inicio del mismo.
r+	Abre el archivo en modo de lectura y escritura empezando por la parte inicial del archivo. El puntero es posicionado al inicio del archivo.
w	Abre el archivo en modo de escritura, empezando por la parte inicial. Si el archivo ya existe, elimina su contenido (o sea lo vuelve a crear). Si no existe intenta crearlo.
w+	Abre el archivo en modo de lectura y escritura, empezando por la parte inicial. Si el archivo ya existe, elimina su contenido (o sea lo vuelve a crear). Si no existe intenta crearlo.
a	Abre el archivo en modo adjunción (pero sólo escritura). Abre el archivo para adjuntar información empezando a escribir por la parte final. Si no existe intenta crearlo.

a+	Abre el archivo en modo adjunción (pero soporta lectura y escritura). Abre el archivo para adjuntar información empezando a escribir por la parte final. Si no existe intenta crearlo.
x	Crea y abre el archivo en modo escritura.Si el archivo existe la funcion fopen() fallará y será generado un error de nivel E_WARNING
x+	Crea y abre el archivo en modo escritura.Si el archivo existe, la función fopen() fallará y será generado un error de nivel E_WARNING

Ejemplo completo del uso de fopen()

```
<?php

$archivo = "archivo.txt";


if ( fopen($archivo, r) )

{

	echo "Pudo abrir el archivo";

}

else

{

	echo "No es posible abrir el archivo";

}


?>
```


Función fclose()

Es importante que al terminar de hacer las operaciones con archivos se cierren los mismos. Para esto utilizamos la función fclose().

fclose(apuntador del archivo).

Ejemplo completo del uso de fclose()

```
<?php

$archivo = "archivo.txt";

@ $fp = fopen($archivo, r);

if ( !$fp )
{
	echo "No es posible abrir el archivo";
}
else
{
	echo "Pudo abrir el archivo";
	fclose($fp);
}

?>
```


Control de Errores

Operador Arroba "@"

El signo arroba (@) es un operador de control de error.

Su finalidad es ignorar los mensajes de error que pudiera arrojar PHP en el momento de la ejecución de una sentencia.

El mismo es colocado al comienzo de una expresión en PHP.

Luego, cualquier mensaje de error que pudiera generarse a causa de esa expresión será ignorado.

Captura de mensajes de error

Estos mensajes de error serán almacenados en la variable $php_errormsg (Siempre y cuando la característica track_errors esté habilitada).

Ejemplo completo del carácter especial “@”

```
<?php

    if (@ !file ('archivo_que_no_existe') )

        die ("La apertura de archivo ha fallado:<BR>". $php_errormsg);

?>
```

Nota:

La función file lee un archivo de texto entero y lo almacena en un arreglo. Al no existir dicho archivo de texto, PHP arroja un mensaje de error, el cual se captura con el carácter especial arroba.

La función die(), imprime un mensaje y da por terminado la ejecución del script. Esta función es equivalente a la función exit().

Lectura de archivos de texto

Introducción

Hasta ahora hemos visto cómo hacer para abrir y cerrar un archivo, pero no hemos realizado ninguna operación sobre el mismo.

Ahora vamos a ver cómo consultar el contenido de un archivo.

Función fpassthru()

Para recorrer un archivo se puede utilizar la función fpassthru() que recibe el parámetro del archivo que se quiere

leer.

fpassthru (apuntador del archivo) imprime todos los datos restantes en el apuntador del archivo hasta llegar a EOF.

Ejemplo completo del uso de fpassthru()

```
<?php
$url = "http://".$_SERVER['HTTP_HOST']."/".@$_SERVER['REQUEST_URI'] ;
$url = str_replace(basename( $url ),"",$url);

$archivo = $url . "php.txt" ;

echo $archivo;
echo "<hr>";

@ $fp = fopen($archivo, r);

if ( !$fp )
{
	echo "No es posible abrir el archivo \n <br>";
}
else
{
	fpassthru ($fp);
	fclose($fp);
}

?>
```

Función fread()

fread(apuntador del archivo, cantidad de caracteres).

Esta función lee el archivo indicado a partir de dónde está posicionado el apuntador hasta la longitud de bytes especificados.

Ejemplo completo del uso de fread()

```
<?php

$archivo = "php.txt" ;

@ $fp = fopen($archivo, r);

if ( !$fp )
{
    echo "No es posible abrir el archivo \n <br>";
}
else
{
    echo "Los diez primeros caracteres son: " . chr(34). fread ($fp, 10) . chr(34);
    fclose($fp);
}

?>
```

Función fgetc()

fgetc(apuntador del archivo).

Esta función permite leer el archivo caracter por caracter a partir de donde está posicionado el apuntador.

Ejemplo completo del uso de fgetc()

```
<?php

$archivo = "php.txt" ;

@ $fp = fopen($archivo, r);

if ( !$fp )
{
	echo "No es posible abrir el archivo \n <br>";
}
else
{
	echo "El primer caracter es: " . chr(34). fgetc ($fp) . chr(34);
	fclose($fp);
}

?>
```

Función feof()

feof(apuntador del archivo).

Esta función verifica si el apuntador a un archivo está al final del mismo.

Ejemplo completo del uso de feof()

```
<?php

$archivo = "php.txt" ;

@ $fp = fopen($archivo, r);

if ( !$fp )
{
	echo "No es posible abrir el archivo \n <br>";
}
else
{
	while ( !feof($fp) )
		echo fgetc ($fp);

	fclose($fp);
}
?>
```

Función fgets()

fgets(apuntador del archivo, [longitud]).

Esta función permite leer una cadena de texto. Comienza desde donde está posicionado el apuntador y como máximo hasta la longitud de bytes especificados. La lectura también termina cuando se encuentra un carácter de nueva línea o se ha llegado al final del archivo.

Función fgetss()

fgetss(apuntador del archivo, longitud, [etiquetas]).

Esta función es igual a la anterior con la diferencia de que intenta eliminar cualquier etiqueta HTML. Esta función es interesante si necesitamos extraer texto de un documento HTML.

Función file()

file(nombre del archivo).

Lee un archivo de texto entero y lo almacena en un arreglo.

Ejemplo:

$archivo = “C:\PHP.txt”;

$vec = file($archivo);

Escritura de archivos de texto

Función fwrite()

fwrite (apuntador del archivo, cadena, [longitud]).

Esta función permite grabar información en un archivo.

```
<?php

@ $fp = fopen("php.txt", a);

if ( !$fp )
{
    echo "No es posible abrir el archivo \n <br>";
}
else
{
    fwrite ($fp,"\n Texto Adicionado \n");

    fclose($fp);
}
```

?>

Función is_writable

Esta función asegura si el archivo existe y puede escribirse sobre él.

is_writable($nombre_archivo))

Moviéndose dentro de un archivo

Todas las funciones que hemos visto hasta el momento para el manejo de archivos contemplan únicamente las operaciones básicas para abrir o crear archivos.

Ahora veremos las operaciones más avanzadas que permitirán moverse dentro de los archivos de texto.

Función rewind()

Retrocede la posición del apuntador del archivo al inicio del mismo. Devuelve TRUE si se llevo a cabo correctamente y FALSE si falló.

bool rewind (resource $gestor)

Define el indicador de posición de archivo para gestor como el comienzo de la secuencia de archivo.

Función fseek()

Realiza una búsqueda sobre el archivo. Esta función recibe dos parámetros: el primero es el apuntador del archivo y el segundo es el número de caracteres donde se quiere posicionarse, o sea, el desplazamiento.

Ejemplo completo del uso de fseek y rewind()

<?php

@ $fp = fopen("php.txt", r);

if (!$fp)

{

```
        echo "No es posible abrir el archivo \n <br>";
}
else
{
        //voy a la posicion 20 del archivo.
        fseek( $fp, 20);

        //Imprimo los 10 caracteres que esten despues
        echo fread ($fp, 10) . "<BR>\n";

        //Vuelvo al principio del archivo.
        rewind ($fp);

        //Imprimo los 20 caracteres anteriores
        echo fread ($fp, 20) . "<BR>\n";

        fclose($fp);
}

?>
```

Funciones de PHP para interactuar con ficheros y directorios

Función copy(). Copiar Archivos

Esta función recibe dos parámetros: el primero es el archivo de origen y el segundo es el archivo destino. Devuelve verdadero si se pudo copiar correctamente; de lo contrario devuelve falso.

```
int copy  ( string $origen  , string $destino  )
```

Realiza una copia de origen a destino. Devuelve TRUE si todo se llevó a cabo correctamente, FALSE en caso de fallo.

```
<?php

$file = 'php.txt';

$newfile = ' php.txt.bak';


if (!copy($file, $newfile)) {

    echo "Error al copiar $file...\n";

}

?>
```

Función rename(). Renombrar Archivos

Esta función renombra un archivo. Funciona exactamente igual a la anterior: devuelve verdadero si se pudo renombrar correctamente y de lo contrario devuelve falso.

bool rename (string $nombre_antiguo , string $nombre_nuevo [, resource $contexto])

Intenta renombrar nombre_antiguo a nombre_nuevo .

```
<?php

rename("php.txt.bak", "xxx.bak");

?
```

Función unlink(). Borrar Archivos

Esta función permite borrar un archivo. Funciona de la misma manera que la anterior: devuelve verdadero si se ha podido borrar correctamente y de lo contrario devuelve falso.

bool unlink (string $nombre_archivo [, resource $contexto])

Elimina nombre_archivo.

Función file_exists()

Esta función permite consultar si el archivo pasado por parámetro existe. Devuelve verdadero si existe y de lo contrario devuelve falso. Es muy útil verificar la existencia del archivo antes de realizar cualquier operación.

bool file_exists (string $nombre_archivo)

Verifica si un archivo o directorio existe.

```
<?php

$nombre_archivo = 'php.txt';

if (file_exists($nombre_archivo)) {

   echo "El archivo $nombre_archivo existe";

} else {

   echo "El archivo $nombre_archivo no existe";

}

?>
```

Función filetime()

Esta función permite consultar la última fecha de acceso al archivo pasado por parámetro.

int fileatime (string $nombre_archivo)

```
<?php

// imprime p.ej. se accedió a un_archivo.txt en: December 29 2002 22:16:23.

$nombre_archivo = 'un_archivo.txt';

if (file_exists($nombre_archivo)) {

   echo "se accedió a $nombre_archivo en: " . date("F d Y H:i:s.", fileatime($nombre_archivo));

}

?>
```

Función filesize()

Esta función permite consultar el tamaño del archivo pasado por parámetro. La medida del valor que devuelve es en bytes.

int filesize (string $nombre_archivo)

```
<?php

// imprime, p.ej. un_archivo.txt: 1024 bytes

$nombre_archivo = 'un_archivo.txt';

echo $nombre_archivo . ': ' . filesize($nombre_archivo) . ' bytes';

?>
```

Función pathinfo()

Esta función permite dividir la información de un archivo. Devuelve un Array asociativo de cuatro elementos compuesto por: la ruta (path), el nombre del directorio, el nombre del archivo y la extinción del archivo.

Por ejemplo, dada la siguiente url:

/home/www/htdocs/prueba/index.html

La función pathinfo() devolverá un vector con la siguiente información:

- Directory name: /home/www/htdocs/prueba
- Base name: index.html
- File extension: html
- File name: index

```
<?php

$pathinfo = pathinfo('/home/www/htdocs/book/chapter10/index.html');

printf("Dir name: %s <br />", $pathinfo['dirname']);

printf("Base name: %s <br />", $pathinfo['basename']);
```

```
printf("Extension: %s <br />", $pathinfo['extension']);

printf("Filename: %s <br />", $pathinfo['filename']);

?>
```

Manejo de Directorios

PHP ofrece un set de funciones para el trabajo con directorios.

Función opendir()

Esta función permite abrir un directorio para realizar operaciones con él, como por ejemplo, listar todo los archivos de un directorio.

resource opendir (string $ruta [, resource $contexto])

```
<?php
$dir = "/pruebas/";

// Abrir un directorio conocido, y proceder a leer sus contenidos
if (is_dir($dir)) {
   if ($gd = opendir($dir)) {
      while (($archivo = readdir($gd)) !== false) {
         echo "nombre de archivo: $archivo : tipo de archivo: " . filetype($dir . $archivo) . "\n";
      }
      closedir($gd);
   }
}
?>
```

Función readdir()

Esta función guarda todos los archivos de un directorio en un arreglo que luego puede ser recorrido el mismo con el ciclo While.

string readdir (resource $gestor_dir)

Función chdir()

Esta función permite cambiar de directorio.

bool chdir (string $directorio)

```
<?php

// directorio actual
echo getcwd() . "\n";

chdir('..');

echo "<br>";

// directorio actual
echo getcwd() . "\n";

?>
```


Resumen

Las constantes son similares a las variables, con la diferencia de que no llevan el signo dólar delante y sólo se puede asignar una vez. Para definir una constantes se utilizará la función "define".

Las constantes predefinidas son creadas por PHP al arrancar, como por ejemplo, PHP_VERSION que contiene la versión de PHP.

La necesidad de obtener datos de un fichero, o bien, de crear uno, es una posibilidad bastante común que surge durante el desarrollo de aplicaciones. PHP provee de una extensa gama de funciones de acceso a ficheros.

fopen (archivo, modo). La función fopen devuelve un valor numérico (indicador de archivo) de tipo integer que

servirá para hacer referencia al archivo abierto. Con esta función se abre un fichero, bien sea local o una dirección de Internet (http:// o ftp://). Los modos r, r+ , w , w+ colocan el puntero de lectura/escritura a principio del fichero, mientras que los modos a , a+ lo colocan al final.

fclose (indicador_archivo). Con esta función se cierra el fichero que marca el indicador de archivo, devuelve TRUE si el fichero se cierra correctamente y FALSE sino se ha podido cerrar.

El operador arroba (@) indica que PHP no tiene que mostrar los mensajes de error ya que se está evaluando esa posibilidad desde el script.

La función fpassthru lee el contenido de un fichero abierto con fopen y envía el resultado a la salida estándar (ventana del navegador si es una aplicación Web). La función fpassthru lee el contenido de un fichero abierto con fopen y envía el resultado a la salida estándar (ventana del navegador, si es una aplicación web).

Si se quiere especificar la cantidad de ficheros que se quiere recuperar se pueden usar fgets o fread. Conjuntamente, se puede usar filesize() (sólo con ficheros locales) para obtener el tamaño del archivo y pasarle el valor como longitud a fread.

Si se quiere acceder a todo el contenido de un fichero y operar con él se puede usar la función file(). Esta función lee el fichero línea por línea y devuelve un arreglo (un elemento por línea).

fwrite y fputs son funciones idénticas. Ambas permiten escribir una línea nueva en el fichero (abierto con fopen). La escritura tendrá lugar sobrescribiendo el contenido o añadiéndolo al final, según el modo usado con fopen.