Funciones

Funciones en PHP

Introducción

En la programación existen a menudo tareas que se componen de trozos de lógica que pueden ser reutilizados en otras partes, no sólo dentro de la misma aplicación, sino también en muchas otras. Por ejemplo, en una aplicación de comercio electrónico puede ser que se necesite una función para validar la dirección de correo electrónico. Esta validación es probable que se necesite aplicar en varias páginas diferentes, como por ejemplo, cuando un nuevo usuario se registra, cuando alguien quiere añadir un comentario del producto, o cuando un visitante se suscriba a un boletín de noticias. La lógica usada para validar una dirección de correo electrónico es bastante compleja, por lo cual sería ideal mantener la lógica en un solo lugar, en vez de literalmente incrustarla en numerosas páginas (cuando se la necesite usar). Especialmente si un día tiene que ser modificada para tener en cuenta un nuevo dominio (como. museo). Afortunadamente, el concepto de incorporar estos procesos repetitivos dentro de una sección con nombre de código y luego invocar este nombre cuando es necesario ha sido durante mucho tiempo una característica clave en cualquier lenguaje de programación. De esta manera, una sección de código se conoce como función, y se le otorga la comodidad de un singular punto de referencia. Si el proceso se define requiere cambios en el futuro, lo que reduce en gran medida tanto la posibilidad de errores de programación y los gastos generales de mantenimiento.

La sintaxis básica para el uso (su ejecución) de una función es:

```
Nombre_funcion (expression_1, expression_2, ..., expression_n)
```

Esto incluye el nombre de la función seguido por una lista separada por comas y entre paréntesis de las expresiones de entrada (que se llaman los argumentos de la función). Las funciones pueden ser llamados con cero o más argumentos en función de sus definiciones.

Cuando PHP se encuentra con una llamada a la función, primero se evalúa cada expresión de la alegación y, a continuación, utiliza estos valores como insumos para la función. Después de la función se ejecuta el valor devuelto. En caso de que lo hubiera sería el resultado de la expresión de la función entera.

Los valores de retorno

Cada llamada a la función es una expresión de PHP, y (al igual que con otras expresiones) sólo hay dos razones por las que se podría querer incluir uno en su código: el valor de retorno o de los efectos secundarios.

El valor de retorno de una función es el valor de la expresión de la función en sí. Se puede hacer exactamente las mismas cosas que con este valor con los resultados de la evaluación de cualquier otra expresión. Por ejemplo, puede asignar a una variable, como en:

```
$my_pi = pi();
```

Funciones para Manejo de Cadenas

Función print()

Muestra una cadena de caracteres.

Es igual a la sentencia echo.

Es indistinto utilizar una u otra.

Función printf()

Muestra una cadena de caracteres.

A diferencia de la sentencia anterior, printf permite pasar formatos particulares.

Su sintaxis: printf(formato, cadena)

Los formatos que se pueden utilizar son:

%: Cada especificación de conversión consiste del signo de porcentaje, seguido por uno o más de los siguientes formatos.

b: presenta al número como binario

c: es presentado como el carácter con dicho valor ASCII

d: es presentado como un número decimal

f: es presentado como un número flotante

o: es presentado como un número octal

s: es presentado como una cadena de caracteres (string)

x: es presentado como un hexadecimal en minúsculas

X: es presentado como un hexadecimal en mayúsculas

Función sprintf()

Esta función hace lo mismos que printf pero permite almacenar los valores en variables.

Ejemplo completo de la función sprintf

```
<?php

$cad = "A continuación se muestran los siguientes valores: ";

$n = 10;

$f = 5.6;
```

```
$res = printf("<P> %s </P><br> Numero entero: %d <br> Numero flotante: %f", $cad, $n, $f);

printf( $res );

?>
```

Función strtoupper() y strtolower ()

Strtoupper: Pasa a mayúsculas una cadena. Devuelve la cadena con todas sus letras en mayúsculas.

Strtolower: Pasa a minúsculas una cadena. Devuelve la cadena con todas sus letras en minúsculas.

```
<?php

$cad = strtoupper( "A continuación se muestran los siguientes valores: " );

?>
```

Función strlen()

Devuelve la cantidad de caracteres que tiene una cadena.

Recibe como argumento una cadena de caracteres.

Devuelve un número entero indicando la longitud de la cadena.

Sintaxis:

```
$cadena = “Curso de PHP”;
echo strlen($cadena) ;        //devuelve: 12
```

Función number_format()

Permite darle formatos a los números.

Devuelve una versión con formato de número.

Esta función acepta uno, dos o cuatro argumentos (pero no tres).

Su estructura:

number_format (número, decimales, punto decimal, separador de miles);

Ejemplo completo de la función number_format

```
<?php

$num = 154643.364554;
echo number_format($num,2,',','.') ;        //devuelve:

?>
```

Función substr()

Permite extraer o cortar una cadena de texto.

Devuelve parte de una cadena.

Su sintaxis: substr (cadena, comienzo, longitud)

Ejemplo completo de la función substr

```
<?php

echo substr('Curso de PHP', 1)              . "<BR>\n";  // urso de PHP
```

```
echo substr('Curso de PHP', 1, 3)        . "<BR>\n";  // urs

echo substr('Curso de PHP', 0, 5)        . "<BR>\n";  // Curso

echo substr('Curso de PHP', 0, 12)       . "<BR>\n";  // Curso de PHP

echo substr('Curso de PHP', -3)             . "<BR>\n";  // PHP

echo substr('Curso de PHP', -3,2)        . "<BR>\n";  // PH



?>
```

Función rtrim()

Elimina el espacio en blanco (o más caracteres) del final de una cadena.

string rtrim (string cadena [, string lista_caracteres])

Como opcional se le pueden indicar una lista de caracteres que se desean eliminar:

rtrim($texto, " \t.");

Función ltrim()

Elimina el espacio en blanco (o más caracteres) del principio de una cadena

string ltrim (string cadena [, string lista_caracteres])

Función trim()

Elimina espacios en blanco (u otros caracteres) del principio y final de una cadena.

string trim (string cadena [, string lista_caracteres])

Función ereg()

Busca una cadena de caracteres dentro de otra.

La búsqueda diferencia mayúsculas y minúsculas.

Devuelve verdadero si encontró alguna coincidencia, o falso si no encontró coincidencias.

También devuelve falso si ocurrió algún error.

Función eregi()

Es igual a la función anterior pero no distingue entre mayúsculas y minúsculas.

Función split()

Esta función se utiliza para dividir cadenas con un separador previamente definido.

El resultado de la división de la cadena se almacena en un arreglo.

Ejemplo completo de la función split

```
<?php

$separador = " ";

$vec = split($separador, "Curso de PHP" );

while ( list($i, $v) = each($vec) )

	echo $v ."<br>\n" ;

?>
```

Nota: Ver función similar explode()

Función addslashes()

Esta función devuelve una cadena con barras invertidas delante de los caracteres que necesitan escaparse en situaciones como consultas de bases de datos, etc. Los caracteres que se escapan son la comilla simple ('), comilla doble ("), barra invertida (\) y NUL (el byte NULL).

```
<?php
```

```
$str = "Mi apellido es D’Lorenzo";

// La salida sera:: Mi apellido es D\'Lorenzo
echo addslashes($str);
?>
```

Funciones de Redondeo

Función Round()

Redondea un valor flotante.

Si el último decimal es 5 redondea hacia arriba. De lo contrario, redondea hacia abajo.

round (float val [, int precision])

Función Ceil()

Redondea fracciones hacia arriba.

float ceil (float valor)

Funciones para Tipos de Datos

Conversión de tipos de datos

Intval: Para obtener el valor entero de una variable

floatval: Para obtener el valor flotante de una variable

strval: Para obtener el valor de cadena de una variable

Consultar tipos de datos

isset(): Determina si una variable está definida

unset(): Remueve una variable dada

gettype():Obtiene el tipo de una variable

settype():Define el tipo de una variable

empty():Determina si una variable está vacía

is_integer(), is_double(), is_string()

intval(), doubleval(), strval()

is_null(): Encuentra si una variable es NULL

is_numeric: Encuentra si una variable es un número o una cadena numérica

Funciones definidas por el usuario

Definición

Aunque gran variedad de bibliotecas de funciones nativas de PHP son un gran beneficio para cualquier programador que busca a evitar reinventar la rueda, tarde o temprano tendrá que ir más allá de lo que se le ofrece en la distribución estándar. Esto significa que tendrá que crear funciones personalizadas, o sea, funciones definidas por el mismo programador. Para ello, tendrá que definir una función utilizando la sintaxis que soporta PHP.

El bloque de código que representa una función debe estar delimitado por los caracteres **{** y **}**.

Su estructura es:

```
function nombre( argumento 1, . . . ,argumento n)
{
    …
    codigo php
    ...
}
```

A menudo resulta útil para pasar los datos a una función. Como ejemplo se creará una función que calcula el costo total de un artículo mediante la determinación de su impuesto sobre las ventas y después se añadirá esa cantidad al precio:

```
function calcVentasImpuestos($precio, $impuesto)
{
                    $total = $precio + ($precio * $impuesto);
```

```
                    echo "El totalt: $total";
}
```

Esta función acepta dos parámetros denominados precio e impuesto. Aunque se pretende que estos parámetros sean reales (con decimales) debido a la debilidad de PHP, nada le impide pasar las variables de cualquier tipo de datos. Pero el resultado podría no ser lo que se espera. Además, se le permite definir pocos o muchos parámetros según considere necesario; no hay limitaciones impuestas por el lenguaje en este sentido.

Una vez definida la función, se puede invocar la misma como se demuestra a continuación:

```
calcVentasImpuestos(15.00, .075);
```

Características

- Las funciones son código escrito que puede ser utilizado las veces que se desee desde cualquier otro programa.
- Permiten escribir un programa en forma modular.
- Son de gran ayuda para coordinar el trabajo en equipo.
- Los nombres de las funciones no son sensibles de las mayúsculas y minúsculas.
- PHP no soporta la redefinición de funciones.

Parámetros

Lista de variables y/o constantes separadas por comas.

Los parámetros son los argumentos que recibe la función.

La cantidad de argumentos pueden ser varios o ninguno.

Parámetros por valor:

Son los parámetros por defecto en el uso de funciones. Significa que si dentro de la función se modifican los valores de los parámetros no se ven modificados fuera de ella.

Parámetros por Referencia

Significa que si dentro de la función se modifican los valores de los parámetros se ven modificados fuera de ella. Para ello es necesario pasar los nombres de los parámetros anteponiendo un ampersand (&).

Parámetros por Defecto

Los valores por defecto se pueden asignar a los argumentos de entrada de la función. Ejemplo: **function calcularValor($parametro = 100)**

Cuando se pasan valores en la definición de la función, estos se asignarán automáticamente al argumento si no se proporciona ningún otro valor. Revisar el ejemplo del impuesto sobre las ventas. A continuación, podría asignar a $impuesto al valor por defecto de 21 por ciento:

```
function calcVentasImpuestos($precio, $impuesto = 21)
{
    $total = $precio + ($precio * $impuesto);
    echo "El totalt: $total";
}
```

Valores de Retornos

A menudo, simplemente confiar en una función para hacer algo no es suficiente. El resultado de una secuencia de comandos podría depender del resultado de la función o en los cambios en los datos resultantes de su ejecución. Sin embargo, el ámbito de las variables impide que la información se transmita fácilmente desde el cuerpo de la función al código donde fue invocada la misma. Para solucionar esto se puede hacer que la función retorne un dato a través de la instrucción **return()**.

La instrucción **return()** no devuelve ningún valor resultante.

```
function calcVentasImpuestos($precio, $impuesto = 21)
{
    $total = $precio + ($precio * $impuesto);
    return $total;
}
```

Como alternativa, puede ser devolver el cálculo directamente incluso sin asignarla a $total, así:

```
function calcVentasImpuestos($precio, $impuesto = 21)
{
    return $precio + ($precio * $impuesto);
}
```

Características de return()

- Los valores se retornan utilizando la instrucción return.

- Pueden retornarse cualquier tipo de valor, inclusive arreglos.
- Se utiliza únicamente desde adentro del código de una funcion
- return() detiene la ejecución de la función desde donde se llame para retornar el valor que se determine.

Ejemplos

Ejemplo Completo de uso de funciones

```
<?php

$num1 = 10;

$num1 = 15;



function suma($a, $b)

{

	$resultado = $a + $b ;

	return ( $resultado );


}


echo "El resultado de 22 + 54 es: " . suma(22,54) . " <br>\n" ;


?>
```


Variables de ambiente

El principio básico que rige las variables en función de los órganos es la siguiente: cada función es su propio pequeño mundo. Esto es, salvo algunas declaraciones especiales, el significado de un nombre de variable dentro de una función que no tiene nada que ver con el significado de ese nombre en otros lugares. Esta es una

característica, por lo que el comportamiento es independiente del contexto. Si no fuera por este tipo de determinación del alcance podría perder mucho tiempo persiguiendo a los errores causados por el uso del mismo nombre de variable en diferentes partes de su código.

Los únicos valores de variables a los que se puede tener acceso por fuera de una función son variables de parámetros formales (que se han copiado de los valores de los parámetros reales), además de las variables asignadas dentro de la función. Esto significa que se puede utilizar las variables locales dentro de una función sin preocuparse por sus efectos sobre el mundo exterior.

Definición

El ámbito es el contexto en el cual la variable tiene validez.

Si una variable se define dentro de una función, entonces no se la podrá utilizar fuera de ella.

Ejemplo Completo de uso de ámbito de variables

```
<?php

$numero = 123;



function ambito()

{

    $numero = 567;

    echo $numero . "<br>\n" ;

}

ambito();

echo $numero . "<br>\n" ;
```

?>

Variables globales

La variables definidas como globales tienen validez en cualquier contexto.

El alcance de una variable definida dentro de una función es local por defecto, lo que significa que, como se explicó en la sección anterior, no tiene relación con el significado de las variables fuera de la función. El uso de la declaración **global** puede informar a **PHP** que desea un nombre de variable a significar lo mismo que lo hace en el contexto fuera de la función. La sintaxis de esta declaración es simplemente la palabra global, seguida por una lista delimitada de variables separadas por comas que deben ser tratadas de esa manera.

Ejemplo Completo de uso de variables globales

```
<?php

$num1 = 10;

$num1 = 15;



function promedio()

{

    global      $num1 ;

    global      $num1 ;


    $promedio = ( $num1 + $num2 ) / 2;


    echo $promedio . "<br>\n" ;


}


echo "El promedio entre dos numeros es: <br>\n" ;
```

```
promedio();


?>
```

Variables de entorno

Introducción

PHP ofrece numerosas variables predefinidas a las que se puede acceder desde cualquier lugar y proporcionan una gran cantidad de información específica sobre el entorno. Se pueden utilizar estas variables para obtener los detalles de la sesión de usuario actual, el sistema operativo y otros datos del entorno.

Características

- Se utilizan para obtener información acerca del servidor.
- El servidor usa estas variables de entornos como argumentos de líneas de comandos.
- Estas variables son cargadas cuando el servidor ejecuta el programa GATEWAY.

Función getenv()

Mediante esta función se puede acceder a las variables de entorno.

Su estructura: getenv(nombre de variable)

Ejemplo de variables de entorno

```
<?php


echo "IP: " . getenv("REMOTE_ADDR") . "<br>\n" ;

echo "PATH: " . getenv("DOCUMENT_ROOT") . "<br>\n" ;

echo "SERVER NAME: " . getenv("SERVER_SOFTWARE") . "<br>\n" ;

echo "BROWSER: " . getenv("HTTP_USER_AGENT") . "<br>\n" ;

?>
```

El Arreglo $_SERVER

$_SERVER es un arreglo (vector) interno de PHP, en el cual se almacena información del servidor.

Contiene las variables definidas por el servidor Web o directamente relacionadas con el entorno en donde el script se está ejecutando. Es análoga a la antigua matriz $HTTP_SERVER_VARS (la cual está todavía disponible aunque no se use).

El siguiente ejemplo recorre el Arreglo e imprime en pantalla el contenido de cada uno de sus valores:

```
<?

for each ( $_SERVER as $i=>$v )
    echo “<b> $i </b>: $_SERVER[$i] <br>”;

?>
```

Esto devuelve una lista de variables similar a la siguiente:

HTTP_HOST => localhost

HTTP_USER_AGENT => Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.1.6)

Gecko/20091201 Firefox/3.5.6 (.NET CLR 3.5.30729) FirePHP/0.3

HTTP_ACCEPT => text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

HTTP_ACCEPT_LANGUAGE => en-us,en;q=0.5

HTTP_ACCEPT_ENCODING => gzip,deflate

HTTP_ACCEPT_CHARSET => ISO-8859-1,utf-8;q=0.7,*;q=0.7

HTTP_KEEP_ALIVE => 300

HTTP_CONNECTION => keep-alive

HTTP_REFERER => http://localhost/chapter03/

HTTP_COOKIE => PHPSESSID=205jm6q0lcj867h8p05umfthm7

PATH => C:\php5212\;C:\Ruby\bin;C:\Program Files\Windows Resource

Kits\Tools\;C:\WINDOWS\system32;C:\WINDOWS;C:\mysql\bin;C:\Program

Files\Java\jdk1.6.0_14\bin;C:\php\PEAR;C:\Program Files\GTK2-Runtime\bin;C:\Program

Files\jEdit;C:\libxslt\bin;C:\libxml2\bin;C:\apache-ant-1.7.1\bin

SystemRoot => C:\WINDOWS

COMSPEC => C:\WINDOWS\system32\cmd.exe

PATHEXT => .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.RB;.RBW

WINDIR => C:\WINDOWS

SERVER_SIGNATURE =>

SERVER_SOFTWARE => Apache/2.2.11 (Win32) PHP/5.2.12

SERVER_NAME => localhost

SERVER_ADDR => 127.0.0.1

SERVER_PORT => 80

REMOTE_ADDR => 127.0.0.1

DOCUMENT_ROOT => C:/apache/htdocs/beginningphpandmysql_4e

SERVER_ADMIN => admin@localhost

SCRIPT_FILENAME => C:/apache/htdocs/beginningphpandmysql_4e/chapter03/server-superglobal.php

REMOTE_PORT => 4298

GATEWAY_INTERFACE => CGI/1.1

SERVER_PROTOCOL => HTTP/1.1

REQUEST_METHOD => GET

QUERY_STRING =>

REQUEST_URI => /chapter03/server-superglobal.php

SCRIPT_NAME => /chapter03/server-superglobal.php

PHP_SELF => /chapter03/server-superglobal.php

REQUEST_TIME => 1262728260

Resumen

Las funciones para el manejo de cadenas son muy importantes en PHP. Normalmente, es necesario enviar datos desde un programa a otro. En otras ocasiones, los datos serán procesados por bases de datos y en otras, en cambio, los datos deberán ser procesados de una manera determinada.

Este capítulo se concentra en uno de los elementos básicos de los lenguajes de programación: la reutilización a través de la programación funcional. Cuando se trabaja en el desarrollo de una aplicación es común que surja la necesidad de ejecutar un mismo bloque de código en diferentes partes de la aplicación. Una función es un bloque de código al que se le pasa una serie de parámetros y devuelve un valor. La utilización de funciones es muy útil para desarrollar el concepto de “Caja Negra”.

Las variables de ambientes definen el contexto en el cual la misma tiene validez, ya sea en forma global o sólo dentro de una función.

Las variables de entorno permiten obtener información acerca del servidor Web. Es posible consultar una lista de todas las variables de entorno usando phpinfo(). Para poder obtener el valor de una variable de entorno se utiliza la función getenv().