Variables; son "contenedores" para el almacenamiento de información. Creación (Declarar) Las variables de PHP, En PHP, una variable comienza con el signo $, seguido del nombre de la variable:

Ejemplo:

<?php

$txt = "Hello world!";

$x = 5;

$y = 10.5;

?>

Después de la ejecución de las afirmaciones anteriores, la variable $ txt contendrá el valor ¡Hola mundo! , La variable $ x contendrá el valor 5 , y la variable $ y contendrá el valor 10.5 .

Nota: Cuando se asigna un valor a una variable de texto, poner comillas alrededor del valor.

Nota: A diferencia de otros lenguajes de programación, PHP tiene ningún mandato para declarar una variable. Se creó el primer momento en que se asigna un valor a la misma.

Piense de variables como recipientes para el almacenamiento de datos.

Las variables de PHP

Una variable puede tener un nombre corto (como X e Y) o un nombre más descriptivo (edad, carname, total\_volume).

Reglas para las variables de PHP:

Una variable comienza con el signo $, seguido del nombre de la variable

Un nombre de variable debe comenzar con una letra o el carácter de subrayado

Un nombre de variable no puede comenzar con un número

Un nombre de variable sólo puede contener caracteres y guiones alfanuméricos (AZ, 0-9 y \_)

Los nombres de variables entre mayúsculas y minúsculas ($ $ edad y la edad son dos variables diferentes)

Recuerde que los nombres de las variables de PHP son mayúsculas y minúsculas!

Variables de salida

La sentencia echo de PHP a menudo se utiliza para enviar datos a la pantalla.

El siguiente ejemplo mostrará cómo texto de salida y una variable:

Ejemplo

<?php

$txt = "W3Schools.com";

echo "I love $txt!";

?>

Ejecutar ejemplo »

El siguiente ejemplo producirá la misma salida que el ejemplo anterior:

Ejemplo

<?php

$txt = "W3Schools.com";

echo "I love " . $txt . "!";

?>

Ejecutar ejemplo »

El siguiente ejemplo es la salida la suma de dos variables:

Ejemplo

<?php

$x = 5;

$y = 4;

echo $x + $y;

?>

Ejecutar ejemplo »

Nota: Usted aprenderá más sobre la declaración de eco y la forma de salida de datos a la pantalla en el siguiente capítulo.

PHP es un lenguaje de programación relajado Typed

En el ejemplo anterior, observe que no teníamos que contar PHP que tipo de datos es la variable.

PHP convierte automáticamente la variable para el tipo de datos correcto, en función de su valor.

En otros lenguajes como C, C ++ y Java, el programador debe declarar el nombre y el tipo de la variable antes de usarla.

Las variables de PHP Alcance

En PHP, las variables se pueden declarar en cualquier parte del guión.

El alcance de una variable es la parte de la secuencia de comandos, donde se puede hacer referencia / usa la variable.

PHP tiene tres ámbitos variables diferentes:

local

global

estático

Alcance global y local

Una variable declarada fuera de una función tiene un alcance global y sólo se puede acceder fuera de una función:

Ejemplo

<?php

$x = 5; // global scope

function myTest() {

// using x inside this function will generate an error

echo "<p>Variable x inside function is: $x</p>";

}

myTest();

echo "<p>Variable x outside function is: $x</p>";

?>

Ejecutar ejemplo »

Una variable declarada dentro de una función tiene un alcance local y sólo se puede acceder dentro de esa función:

Ejemplo

<?php

function myTest() {

$x = 5; // local scope

echo "<p>Variable x inside function is: $x</p>";

}

myTest();

// using x outside the function will generate an error

echo "<p>Variable x outside function is: $x</p>";

?>

Ejecutar ejemplo »

Puede tener variables locales con el mismo nombre en diferentes funciones, ya que las variables locales sólo son reconocidos por la función en la que se declaran.

PHP Palabra Clave mundial

La palabra clave global se utiliza para acceder a una variable global dentro de una función.

Para ello, utilice la palabra clave global antes de las variables (dentro de la función):

Ejemplo

<?php

$x = 5;

$y = 10;

function myTest() {

global $x, $y;

$y = $x + $y;

}

myTest();

echo $y; // outputs 15

?>

Ejecutar ejemplo »

PHP también almacena todas las variables globales en una matriz llamada $ GLOBALS [ Índice ]. El índice contiene el nombre de la variable. Esta matriz también se puede acceder desde dentro de las funciones y se puede utilizar para actualizar las variables globales directamente.

El ejemplo anterior se puede reescribir como esto:

Ejemplo

<?php

$x = 5;

$y = 10;

function myTest() {

$GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];

}

myTest();

echo $y; // outputs 15

?>

Ejecutar ejemplo »

PHP la palabra clave static

Normalmente, cuando se ha completado una función / ejecutada, todas sus variables se eliminan. Sin embargo, a veces queremos una variable local no va a eliminar. Lo necesitamos para un trabajo más.

Para ello, utilice la estática de palabras clave cuando se declara primero la variable:

Ejemplo

<?php

function myTest() {

static $x = 0;

echo $x;

$x++;

}

myTest();

myTest();

myTest();

?>

Ejecutar ejemplo »

Entonces, cada vez que la función se llama, esa variable todavía tendrá la información que contenía desde la última vez que la función se llama.

Nota: La variable sigue siendo local a la función.

Las constantes son como las variables, excepto que una vez que se definen no pueden ser cambiados o indefinido.

**PHP Constantes**

Una constante es un identificador (nombre) para un valor simple. El valor no se puede cambiar durante la secuencia de comandos.

Un nombre válido constante comienza con una letra o un guión bajo (sin signo $ antes del nombre de constante).

Nota: A diferencia de las variables, las constantes son automáticamente mundial a través de toda la secuencia de comandos.

Crear una constante de PHP

Para crear una constante, utilizar la función de definir ().

Sintaxis

define(name, value, case-insensitive)

parámetros:

Nombre : especifica el nombre de la constante

valor : Especifica el valor de la constante

mayúsculas y minúsculas : especifica si el nombre de la constante debería ser sensible a las mayúsculas. Predeterminado es falso

El siguiente ejemplo crea una constante con un caso sensible nombre:

Ejemplo

<?php

define("GREETING", "Welcome to W3Schools.com!");

echo GREETING;

?>

Ejecutar ejemplo »

El ejemplo a continuación crea una constante con un caso-insensible nombre:

Ejemplo

<?php

define("GREETING", "Welcome to W3Schools.com!", true);

echo greeting;

?>

Ejecutar ejemplo »

Las constantes son globales

Las constantes son automáticamente mundial y se pueden utilizar a través de toda la secuencia de comandos.

El ejemplo siguiente utiliza una constante dentro de una función, incluso si está definida fuera de la función:

Ejemplo

<?php

define("GREETING", "Welcome to W3Schools.com!");

function myTest() {

echo GREETING;

}

myTest();

?>

**Clases**

En este ejemplo, primero definimos una clase base y una extensión de la misma. La clase base describe una verdura en general, si es comestible o no y cuál es su color. La subclase Espinaca añade un método para cocinarla y otro para averiguar si está cocinada.

Ejemplo #1 classes.inc

<?php

// clase base con propiedades y métodos miembro

class Verdura {

var $comestible;

var $color;

function Verdura($comestible, $color="verde")

{

$this->comestible = $comestible;

$this->color = $color;

}

function es\_comestible()

{

return $this->comestible;

}

function qué\_color()

{

return $this->color;

}

} // fin de la clase Verdura

// ampliar la clase base

class Espinaca extends Verdura {

var $concinada = false;

function Espinaca()

{

$this->Verdura(true, "verde");

}

function cocinarla()

{

$this->concinada = true;

}

function está\_cocinada()

{

return $this->concinada;

}

} // fin de la clase Espinaca

?>

Entonces instanciamos dos objetos de estas clases e imprimimos la información sobre ellos, incluida su linaje de clases. También definimos algunas funciones de utilidad, prinpipalmente para tener una impresión atractiva de las variables.

Ejemplo #2 test\_script.php

<pre>

<?php

include "classes.inc";

// funciones de utilidad

function imprimir\_vars($obj)

{

foreach (get\_object\_vars($obj) as $prop => $val) {

echo "\t$prop = $val\n";

}

}

function imprimir\_métodos($obj)

{

$arr = get\_class\_methods(get\_class($obj));

foreach ($arr as $método) {

echo "\tfunción $método()\n";

}

}

function linaje\_de\_clases($obj, $clase)

{

if (is\_subclass\_of($GLOBALS[$obj], $clase)) {

echo "El objeto $obj pertenece a la clase " . get\_class($GLOBALS[$obj]);

echo ", una subclase de $clase\n";

} else {

echo "El objeto $obj no pertenece a una subclase de $clase\n";

}

}

// instancias 2 objetos

$vegetariano = new Verdura(true, "blue");

$frondoso = new Espinaca();

// imprimir información sobre los objetos

echo "vegetariano: CLASE " . get\_class($vegetariano) . "\n";

echo "frondoso: CLASE " . get\_class($frondoso);

echo ", MADRE " . get\_parent\_class($frondoso) . "\n";

// mostrar las propiedades de vegetariano

echo "\nvegetariano: Propiedades\n";

imprimir\_vars($vegetariano);

// y los métodos de frondoso

echo "\nfrondoso: Métodos\n";

imprimir\_métodos($frondoso);

echo "\nLinaje:\n";

linaje\_de\_clases("frondoso", "Espinaca");

linaje\_de\_clases("frondoso", "Verdura");

?>

</pre>

Una cosa importanete a observar en el ejemplo anterior es que el objeto $frondoso es una instancia de la clase Espinaca que es una subclase de Verdura, por lo tanto, la última parte del script anterior imprimirá:

[...]

Linaje:

El objeto frondoso no pertenece a una subclase de Espinaca

El objeto frondoso pertenece a la clase Espinaca, una subclase de Verdura

**PHP 5 Funciones**

El verdadero poder de PHP viene de sus funciones; que cuenta con más de 1000 funciones incorporadas.

PHP funciones definidas por el usuario

Además de las funciones de PHP incorporadas, podemos crear nuestras propias funciones.

Una función es un bloque de instrucciones que se pueden utilizar en varias ocasiones en un programa.

Una función no se ejecutará inmediatamente cuando se carga una página.

Una función se ejecutará mediante una llamada a la función.

Crear una función definida por el usuario en PHP

Una declaración de la función definida por el usuario comienza con la palabra "función":

Sintaxis

function functionName() {

code to be executed;

}

Nota: Un nombre de función puede comenzar con una letra o un guión bajo (no es un número).

Consejo: Dar la función de un nombre que refleja lo que hace la función!

Los nombres de funciones no son mayúsculas y minúsculas.

En el siguiente ejemplo, creamos una función llamada "writeMsg ()". La llave de apertura rizado ({) indica el comienzo del código de función y de la llave de cierre (}) indica el final de la función. La función da salida a "¡Hola mundo!". Para llamar a la función, simplemente escriba su nombre:

Ejemplo

<?php

function writeMsg() {

echo "Hello world!";

}

writeMsg(); // call the function

?>

Ejecutar ejemplo »

Argumentos de función de PHP

La información puede suministrarse a las funciones a través de argumentos. Un argumento es como una variable.

Los argumentos se especifican después de que el nombre de la función, dentro de los paréntesis. Puede añadir tantos argumentos como desee, sólo les separan con una coma.

El siguiente ejemplo tiene una función con un argumento ($ fname). Cuando la función de familyName () se llama, también pasar a lo largo de un nombre (por ejemplo, Jani), y el nombre se utiliza dentro de la función, que da salida a varios nombres diferentes, pero un igual apellidos:

Ejemplo

<?php

function familyName($fname) {

echo "$fname Refsnes.<br>";

}

familyName("Jani");

familyName("Hege");

familyName("Stale");

familyName("Kai Jim");

familyName("Borge");

?>

Ejecutar ejemplo »

El siguiente ejemplo tiene una función con dos argumentos ($ fname y año $):

Ejemplo

<?php

function familyName($fname, $year) {

echo "$fname Refsnes. Born in $year <br>";

}

familyName("Hege", "1975");

familyName("Stale", "1978");

familyName("Kai Jim", "1983");

?>

Ejecutar ejemplo »

PHP por defecto argumento de valor

El siguiente ejemplo muestra cómo utilizar un parámetro predeterminado. Si llamamos a la función setHeight () sin argumentos que toma el valor por defecto como argumento:

Ejemplo

<?php

function setHeight($minheight = 50) {

echo "The height is : $minheight <br>";

}

setHeight(350);

setHeight(); // will use the default value of 50

setHeight(135);

setHeight(80);

?>

Ejecutar ejemplo »

Funciones PHP - valores que vuelven

Para permitir que una función devuelve un valor, utilice la instrucción de retorno:

Ejemplo

<?php

function sum($x, $y) {

$z = $x + $y;

return $z;

}

echo "5 + 10 = " . sum(5, 10) . "<br>";

echo "7 + 13 = " . sum(7, 13) . "<br>";

echo "2 + 4 = " . sum(2, 4);

?>

**Objetos**

Ya que sabemos algo de teoría vamos a ver algo de código para terminar de entender todo lo que hemos visto hasta ahora. Vamos a aplicar OOP en PHP declarando una clase y luego creando un objeto.

Lo primero que necesitamos es una clase, sin estas no existirían los objetos. Para declarar una clase utilizamos la palabra reservada class seguido por el nombre de la clase y dentro de paréntesis todas sus propiedades y métodos.

<?php

class Carro {

var $color;

var $numero\_puertas;

var $marca;

var $gasolina;

function llenarTanque($gasolina\_nueva){

$this->gasolina = $this->gasolina + $gasolina\_nueva;

}

function acelerar(){

$this->gasolina = $this->gasolina - 1;

return 'Gasolina restante: '.$this->gasolina;

}

}

?>

Ahora que tenemos la clase declarada, podemos crear uno o varios objetos de esta. La acción de crear un objeto es instanciar. Para aprender como hacer esto utilizaremos la clase que creamos de Carro. Crear un objeto en PHP es muy sencillo y se hace utilizando la palabra reservada new. Los objetos se deben guardar en variables, por lo tanto para crear un objeto escribiríamos la siguiente linea $carro = new Carro();. Ahora veamos como se hace en un código completo.

<?php

class Carro {

var $color;

var $numero\_puertas;

var $marca;

var $gasolina = 0;

function llenarTanque($gasolina\_nueva){

$this->gasolina = $this->gasolina + $gasolina\_nueva;

}

function acelerar(){

if ($this->gasolina > 0){

$this->gasolina = $this->gasolina - 1;

return 'Gasolina restante: '.$this->gasolina;

}

}

}

$carro = new Carro(); // Instanciamos la clase Carro

$carro->color = 'Rojo'; // Llenamos algunas de las propiedades

$carro->marca = 'Honda';

$carro->numero\_puertas = 4;

$carro->llenarTanque(10); // utilizamos los metodos

$carro->acelerar();

$carro->acelerar();

$carro->acelerar();

?>

Como podemos observar para hacer uso de las propiedades o métodos de un objeto lo hacemos con -> indicando que propiedad o método deseamos ejecutar.

**Estructuras de control**

* Introducción
* if
* else
* elseif/else if
* Sintaxis alternativa de estructuras de control
* while
* do-while
* for
* foreach
* break
* continue
* switch
* declare
* return
* require
* include
* require\_once
* include\_once
* goto

**Patron de diseño MVC**

De manera genérica, los componentes de MVC se podrían definir como sigue:

El Modelo: Es la representación de la información con la cual el sistema opera, por lo tanto gestiona todos los accesos a dicha información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan descrito en las especificaciones de la aplicación (lógica de negocio). Envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario). Las peticiones de acceso o manipulación de información llegan al 'modelo' a través del 'controlador'.12

El Controlador: Responde a eventos (usualmente acciones del usuario) e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). También puede enviar comandos a su 'vista' asociada si se solicita un cambio en la forma en que se presenta el 'modelo' (por ejemplo, desplazamiento o scroll por un documento o por los diferentes registros de una base de datos), por tanto se podría decir que el 'controlador' hace de intermediario entre la 'vista' y el 'modelo' (véase Middleware).

La Vista: Presenta el 'modelo' (información y lógica de negocio) en un formato adecuado para interactuar (usualmente la interfaz de usuario) por tanto requiere de dicho 'modelo' la información que debe representar como salida.

*Interacción de los componentes*

Aunque se pueden encontrar diferentes implementaciones de MVC, el flujo de control que se sigue generalmente es el siguiente:

El usuario interactúa con la interfaz de usuario de alguna forma (por ejemplo, el usuario pulsa un botón, enlace, etc.)

El controlador recibe (por parte de los objetos de la interfaz-vista) la notificación de la acción solicitada por el usuario. El controlador gestiona el evento que llega, frecuentemente a través de un gestor de eventos (handler) o callback.

El controlador accede al modelo, actualizándolo, posiblemente modificándolo de forma adecuada a la acción solicitada por el usuario (por ejemplo, el controlador actualiza el carro de la compra del usuario). Los controladores complejos están a menudo estructurados usando un patrón de comando que encapsula las acciones y simplifica su extensión.

El controlador delega a los objetos de la vista la tarea de desplegar la interfaz de usuario. La vista obtiene sus datos del modelo para generar la interfaz apropiada para el usuario donde se reflejan los cambios en el modelo (por ejemplo, produce un listado del contenido del carro de la compra). El modelo no debe tener conocimiento directo sobre la vista. Sin embargo, se podría utilizar el patrón Observador para proveer cierta indirección entre el modelo y la vista, permitiendo al modelo notificar a los interesados de cualquier cambio. Un objeto vista puede registrarse con el modelo y esperar a los cambios, pero aun así el modelo en sí mismo sigue sin saber nada de la vista. Este uso del patrón Observador no es posible en las aplicaciones Web puesto que las clases de la vista están desconectadas del modelo y del controlador. En general el controlador no pasa objetos de dominio (el modelo) a la vista aunque puede dar la orden a la vista para que se actualice. Nota: En algunas implementaciones la vista no tiene acceso directo al modelo, dejando que el controlador envíe los datos del modelo a la vista. Por ejemplo en el MVC usado por Apple en su framework Cocoa. Suele citarse como Modelo-Interface-Control, una variación del MVC más puro

La interfaz de usuario espera nuevas interacciones del usuario, comenzando el ciclo nuevamente....

*MVC y bases de datos*

Muchos sistemas informáticos utilizan un Sistema de Gestión de Base de Datos para gestionar los datos que debe utilizar la aplicación; en líneas generales del MVC dicha gestión corresponde al modelo. La unión entre capa de presentación y capa de negocio conocido en el paradigma de la Programación por capas representaría la integración entre la Vista y su correspondiente Controlador de eventos y acceso a datos, MVC no pretende discriminar entre capa de negocio y capa de presentación pero si pretende separar la capa visual gráfica de su correspondiente programación y acceso a datos, algo que mejora el desarrollo y mantenimiento de la Vista y el Controlador en paralelo, ya que ambos cumplen ciclos de vida muy distintos entre sí.