PHP5 Orientado a Objetos

Hola a todos, en esta ocasión les traigo a ustedes un poco de información acerca de las mejoras introducidas en PHP5 las cuales son principalmente la orientación a objetos que está muy bien lograda en esta nueva versión.

Primero debemos entender algunos conceptos de la programación orientada a objetos, comencemos con los constructores.

¿Qué es un constructor?

Un **constructor** es una función que se ejecuta cada vez que se instancia un objeto de una clase. En PHP4 el nombre de dicha función debía ser identifica al nombre de la clase, pero en PHP5, ya existe un nombre unificado para definir un constructor: **__construct**.

```
class ClaseA {
   var $a;

   function __construct() {
      $this->a = true;
   }
}

$Objeto = new ClaseA();
```

Si PHP no encuentra este nombre de función, buscará un método que se llame igual que la clase, tal y como lo hacía en PHP4.

De esta misma manera en PHP5 se incluyeron los destructores, pero ¿sabes que es un destructor?

¿Qué es un destructor?

Pues son muy similares que los constructores. Son funciones que ejecutan tareas que necesitamos hacer cuando un objeto ya no está siendo referenciado por ninguna variable, antes de liberar memoria. El nombre unificado de los destructores es: **__destruct**.

```
class ClaseA {

   var $a;

   function __construct() {
      $this->a = true;
   }

   function __destruct() {
      echo "Objeto Destruido";
   }
}

$Objeto = new ClaseA();
```

Los destructores incluyen un código que se ejecuta cuando un objeto se elimina de la memoria.

Modificadores de acceso

Bueno vamos entrando en calor, otras de las mejoras que se incluyeron en PHP5 son los modificadores (*public, prívate y protected*) de acceso sobre las propiedades y los métodos.

- **Public:** la propiedad o método es accesible desde cualquier ámbito, incluyendo otras claves.
- **Private:** la propiedad o método solo es accesible desde dentro de la clase a la que pertenece.
- **Protected:** la propiedad o método sólo es accesible desde dentro de la clase a la que pertenece o desde cualquier clase que derive de ella.

```
class ClaseA {
   public $a;
   protected $b;
   private $c;

   function __construct() {
      $this->b = 100;
   }

   function GetB() {
      return $this->b;
   }
}

$Objeto = new ClaseA();

echo $Objeto->GetB();
```

En la versión anterior PHP4, todas las variables y las funciones son públicas.

En PHP4, las propiedades se declaraban mediante la palabra reservada **var** (sinónimo de public) y los métodos se definían con su nombre y argumentos.

Así como existen las propiedades de un objeto, también existen las propiedades de una clase (se definen por la palabra **static**), cuyos valores no dependen específicamente de una instancia en particular.

El acceso a los atributos de clase se realiza utilizando el operador **::**, y para eso no es necesario instanciar la clase.

```
class ClaseA {
   static $Propiedad = 15;

   static function ShowValue () {
      echo ClaseA::$Propiedad;
   }
}

ClaseA::ShowValue();
```

Nuevos Métodos

Un Nuevo método que incluye PHP5 es **__call**, que se ejecuta de manera automática en caso de llamar un método que no está definido en la clase.

```
class ClaseA {
   public $a;

   function __construct() {
      $this->a = 100;
   }

   function GetA() {
      return $this->a;
   }

   function __call($metodo, $atributos) {
      echo "El Método $metodo no existe";
   }
}

$Objeto = new ClaseA();
echo $Objeto->GetB();
```

El método **__call** recibe dos argumentos automáticos: el nombre del método inexistente y los parámetros pasados a él en forma de array.

Los métodos **__set** y **__get**, en caso de estar definidos, serán invoados cuando se intente acceder, o modificar, el valor de alguna propiedad no declarada en la clase.

Clonación de objetos

Cuando se crea una instancia de una clase y se asigna esa instancia a otra variable, se dice que se crea un alias del objeto.

```
$ObjetoA = new ClaseA();
$ObjetoB = $ObjetoA;
```

No se duplica el objeto entero como sucedía en PHP4, donde, si queríamos referenciar objetos, debíamos hacerlo de forma explícita.

```
$ObjetoA =& new ClaseA();
$ObjetoB =& $ObjetoA;
```

En PHP5, las variables que representan objetos son referencias a ellos mismos, por lo que no es necesario utilizar el operador **&** (ampersand).

Clases abstractas

Las clases abstractas no pueden ser instanciadas y el único fin que tienen es servir como base para otras clases (que si pueden ser instanciadas). Para definir un método abstracto se hace de la siguiente manera.

```
abstract class Ejemplo {
    public $Propiedad;
    abstract function MetodoAbstracto();
}
```

Interfaces

PHP5 admite el uso de interfaces que se utilizan principalmente para declarar el conjunto de métodos que deberán definir (obligatoriamente) las clases que implementan la interfaz.

Las interfaces son definidos utilizando la palabra reservada interface.

```
interface Ejemplo {
    public function Method1($var);
    public function Method2();
}

class ClaseA implements Ejemplo {
    ...
}
```

Las clases pueden implementar más de una interfaz, separando cada una de ellas mediante el uso del carácter coma (,).

```
class claseA implements interfazA, interfazB, interfazC { ... }
```

Operador InstanceOf

Este operador permite saber si un objeto es una instancia de una determinada clase.

```
include "animales.class.php";

$Obj = new Perro();
$Obj->Nombre = "Valiente";

if($Obj instanceof Perro) echo $Obj->Animal().' es un perro';
elseif($Obj instanceof Gato) echo $Obj->Animal().' es un gato';
else echo $Obj->Animal().' no es ni perro ni gato';
```

Autocarga de clases

Una de las grandes mejoras de PHP es la función **__autoload**, si está definida, es llamada automáticamente al intentar utilizar una clase que no ha sido definida.

```
function __autoload($Class) {
    $File = 'classes/'.$Class.'.class.php';
    if(file_exists($File)) include $File;
    else die("No se encuentra el archive de la clase $Class");
}

$Obj = new MySQL(); //Autoload intentará cargar: /classes/MySQL.class.php
```

Acceso a subobjetos

Por último veremos el acceso a subobjetos, recordemos que en PHP4 era necesario trabajar con variables temporales para poder accede a un objeto instanciado dentro de otro. En PHP esto cambio y ya es posible hacerlo de forma directa.

```
class ClaseA {
   public $Propiedad;

   function __construct() {
      $this->Propiedad = new ClaseB();
   }
}

class ClaseB {
   function Message($Mensaje) {
      echo $Mensaje;
   }
}

$Obj = new ClaseA();

echo $Obj->Propiedad->Mensaje("Hola Mundo");
```

Bueno espero que estos pequeños ejemplos les hayan sido de utilidad y espero que hayan podido aprender un poco más acerca de PHP5, la próxima publicación será sobre como instalar la nueva versión del sistema operativo Ubuntu 9.04 para aquellas personas que les interese conocerlo, probarlo, olvidarse de los virus y porqué no reemplazarlo por su poco estable Windows (98, XP, Vista, 7).

Si gustan profundizar más en el tema les recomiendo que lean el libro de PHP5: Evolución y Madurez de Francisco Minera, muy bueno por cierto.

Un Saludo,

Atentamente

Carlos Santana Roldán

NOTA: *Este tutorial fue creado por Carlos Santana Roldán para MilkZoft, si deseas tomar total o parcialmente el contenido publicado te pedimos nos envíes un mensaje utilizando el formulario de contacto y agregando un enlace a MilkZoft como fuente de los datos (puedes insertar el siguiente código en tu blog o página web).*

Fuente: MilkZoft