Clases y objetos

Aunque PHP no es un lenguaje **orientado a objetos**, sí tiene recursos que permiten definir *clases* y construir *objetos*.

El uso de **clases** y **objetos** no añade ninguna funcionalidad nueva a las posibilidades de PHP. Su verdadera utilidad es la de *hacer la programación de otra manera*, con un código más legible y reutilizable.

Las clases

Una clase no es otra cosa que una especie de *plantilla* en la que se pueden definir una serie de **variables** –que pueden contener valores predefinidos– y un conjunto de **funciones** que pueden ser *invocadas* desde cualquier parte del documento.

La sintaxis es la siguiente:

class *nombre* **{**
....
... definición de variables....
....
.. constructores (opcional)...
....
.. definición de funciones...
....
}

Vayamos por partes.

Dentro de una *clase* podemos *definir las variables* que serán utilizadas por sus *funciones internas* y a las que es posible (no es imprescindible hacerlo) *asignar* valores.

Para definir una variable es **obligatorio** *anteponer* ***var*** a su nombre y en el caso de que tratemos de asignarle un valor, bastará con poner detrás del nombre el signo **=** seguido del valor.

Ni que decir tiene que el nombre de la variable utiliza la sintaxis habitual de PHP y que si los valores asignados son tipo **cadena** tienen que ir *entre comillas*.

var *$pepe="Jose"*

es una sintaxis válida, pero

$pepe="Jose"

no lo es, le falta el **var** y si lo escribimos así nos dará un *error*.

Más adelante hablaremos de los **constructores**, pero dado su carácter *opcional* veamos antes las funciones.

Las **funciones** definidas dentro de las clases tienen una sintaxis **idéntica** al resto de las funciones PHP con una **salvedad** importante:

Siempre que desde una **función** –contenida en una clase– se trate de *invocar* una **variable** definida **en la misma clase** ha de hacerse con la siguiente sintaxis:

$this->*variable*

Prestemos mucha atención. El **$** va siempre **delante** de la palabra **this** y solo se escribe **una vez** y en esa posición.
El nombre de la variable (**que va siempre después** de **->** *no lleva*

El ejemplo más simple

En este ejemplo podemos ver como **utilizar una clase** para definir una *plantilla* que va a multiplicar **siete** por **ocho** y que nos va a devolver el resultado de esa operación cada vez que sea invocada.

```
<?
class Multiplica{
    var $factor1=7;
    var $factor2=8;
    function curratelo(){
        echo $this->factor1*$this->factor2;
       }
}
   $objeto= new Multiplica;
   $objeto->curratelo();
 ?>
```

Ver ejemplo70.php

Invocando varias veces el mismo objeto

En este ejemplo puedes observar cómo al *invocar* dos veces a **$objeto** el valor que nos devuelve es el resultado de la **última llamada** y también como se pueden crear **dos objetos** distintos.

```
<?
class Multiplica{
    var $resultado;
    function curratelo($a,$b){
      $this->resultado=$a*$b;
       }
    function imprimelo(){
        echo $this->resultado,"<br>";
       }
}
   $objeto= new Multiplica;
   $objeto->curratelo(7,3);
   $objeto->curratelo(11,4);
   $objeto1= new Multiplica;
   $objeto1->curratelo(-23,11);
   $objeto->imprimelo();
   $objeto1->imprimelo();
 ?>
```

Ver ejemplo71.php

Recogiendo resultados en un *array*

En este ejemplo vemos cómo se puede *invocar* reiteradamente una **función** utilizando el mismo objeto y como pueden recogerse esos resultados en un *array*.

```
<?
class Multiplica{
    var $resultado;
    var $indice=0;

   function curratelo($a,$b){
      $this->resultado[$this->indice]=$a*$b;
      $this->indice++;
       }

   function imprimelo(){
       foreach($this->resultado as $valor){
            echo $valor,"<br>";
        }
   }

}

   $objeto= new Multiplica;

   $objeto->curratelo(7,3);
   $objeto->curratelo(11,4);
   $objeto->curratelo(-23,11);
   $objeto->imprimelo();
 ?>
```

Ver ejemplo72.php

pegado el ***$***.

¡Observa los ejemplos!

Crear y utilizar objetos

Las **clases** son solo *plantillas* y sus **funciones** no se ejecutan hasta que se les **ordene**.

Para poder utilizarlas primero debemos **crear un objeto** y luego **ejecutar** *sobre ese objeto* la función o funciones que deseemos.

Creación de objetos

Para crear un **objeto** en el que se vaya a utilizar **una clase determinada** debemos usar la siguiente sintaxis:

$nombre* = new *clase

donde **nombre** es una *palabra cualquiera* con la que identificar el **objeto** (como si se tratara de una variable) y **clase** es el **nombre** de una de las clases definidas.

¡Cuidado!

Fíjate que *detrás del nombre de la clase* **no hemos puesto los ()** que suelen utilizarse para invocar las *funciones*.

Utilizando objetos

Una vez definido un **objeto** ya se le pueden aplicar las **funciones** definidas en la **clase**. La sintaxis es la siguiente:

***$nombre*->*funcion*()**

Donde **$nombre** es la *misma variable* utilizada a la hora de **crear** el objeto, el **->** es obligatorio, **funcion** es el nombre de **una de las funciones** definidas en la *clase* invocada y donde los **()** sí **son obligatorios** y además -como ocurría en las demás funciones PHP- puede contener *valores*, *variables*, etcétera separadas por comas.

Reiterando llamadas a objetos

Si hacemos varias *llamadas* a una función utilizando el *mismo objeto* los resultados se van *sobrescribiendo* sobre los de la llamada anterior.

Si queremos conservar *varios resultados* obtenidos de la aplicación de la *misma función* tenemos dos opciones:

- Crear varios objetos
- Utilizar *arrays*

En los ejemplos podemos ver ambos supuestos.

Constructores

Cuando se define *una función* cuyo nombre es *idéntico* al de la *clase* que la contiene recibe el nombre de **constructor**.

Esa función -el **constructor**- se *ejecuta* de forma *automática* en el momento en que se define un *nuevo objeto* (**new**)

Según como esté *definido*, el **constructor** puede ejecutarse de distintas formas:

• *Con los valores predefinidos* en las variables de la *clase*.

• Mediante la asignación de *valores preestablecidos* en los propios parámetros de la función

Ejemplos de uso de un *constructor*

```
<?
class Multiplica{
    var $factor1=7;
    var $factor2=8;
    function Multiplica(){
        print $this->factor1*$this->factor2."<br>";
    }
}

   $objeto= new Multiplica;

 ?>
```

Ver ejemplo73.php

```
<?
class Multiplica{

    var $producto;

    function Multiplica($a=3,$b=7){
        $this->producto=$a*$b;
        print $this->producto."<br>";
    }
}
   $objeto= new Multiplica;

   $objeto->Multiplica(90,47);

   $objeto->Multiplica(47);

   $objeto->Multiplica();


 ?>
```

Ver ejemplo74.php

Un ejemplo más completo

En este ejemplo puedes ver como el **constructor** crea automáticamente los valores de la **primera fila** de la tabla resultante.

```
<?
Class Operaciones {
    var $inicializada=32;
    var $num1;
    var $num2;
    var $suma;
    var $diferencia;
    var $producto;
    var $cociente;
    var $contador=0;
      function Operaciones ($val1=45,$val2=55){
        $this->contador +=1;
        $c=$this->contador;
        $this->num1[$this->contador]=$val1;
        $this->num2[$c]=$val2;
        $this->suma[$c]=$val1+$val2;
        $this->diferencia[$c]=$val1-$val2;
        $this->producto[$c]=$val1*$val2;
        $this->cociente[$c]=$this->inicializada*$val1/$val2;
              }


     function imprime(){
         print "<table align=center border=1>";
         print "<td>Num 1</td><td>num2</td><td>Suma</td>";
         print "<td>Diferencia</td><td>Producto</td><td>Cociente</
td><tr>";
         foreach($this->num1 as $clave=>$valor){
             print "<td align=center>".$valor."</td>";
             print "<td align=center>".$this->num2[$clave]."</td>";
             print "<td align=center>".$this->suma[$clave]."</td>";
             print "<td align=center>".$this->diferencia[$clave]."</td>";
             print "<td align=center>".$this->producto[$clave]."</td>";
             print "<td align=center>".$this->cociente[$clave]."</
td><tr>";
      }
       print "</table>";
```

constructor.

En el ejemplo puedes ver la sintaxis de esta forma en la que se le asignan los valores **3** y **7** a las variables **$a** y **$b**.

En el mismo ejemplo puedes ver también la *utilidad añadida* del **constructor**.

Cuando se le *pasan* valores la función se ejecuta *sin tomar en consideración* los asignados por defecto en la función y cuando se le *pasan* sólo **parte** de esos valores utiliza los **valores recibidos** y para los *no asignados en la llamada* utiliza los valores del **constructor**.

Clases extendidas

PHP también tiene la posibilidad de crear **clases extendidas** cuya *virtud* es poder disponer tanto de *variables y funciones propias* como de *todas las variables y funciones* de la **clase padre**.

Sintaxis de las clases extendidas

Para crear **clases extendidas** se requiere utilizar la siguiente sintaxis:

class *nuev* **extends** *base* **{**
....
... definición de variables....
....
.. constructores (opcional)...
....
.. definición de funciones...
....
}

Como habrás podido deducir *nuev* es el *nombre de la nueva clase* (la **extendida**), *base* es el nombre de la clase **padre** y **extens** es la *palabra clave* que indica a PHP que se trata de una clase extendida.

PHP no permite las *herencias múltiples*. No es posible crear *una clase extendida de otra clase extendida*.

Funciones con Clases y objetos

Existen algunas funciones que pueden resultarte útiles a la hora de trabajar con clases y objetos. Son las siguientes:

method_exists(*obj, func***)**

Comprueba si está definida la función **func** (función y método son sinónimos) en el objeto **obj**.

Devuelve un valor *booleano*. Cierto (true) en el caso de que exista esa función y **falso** en el caso de que no exista.

get_class_vars(*clase***)**

Crea un **array asociativo** cuyos *índices* son los *nombres de las variables* y cuyos valores coinciden con los *valores preasignados* a cada una de esas variables.

En este **array** solo se recogen las variables que han sido *inicializadas* asignándoles un *valor*.

get_class_methods(*clas***)**

Devuelve un *array* conteniendo los valores de todos los **métodos** (funciones) definidas en la *clase* **clas**.

get_object_var(*obj***)**

```
    }
}
$objeto= new Operaciones;

for ($i=1;$i<11;$i++){
    for ($j=1;$j<11;$j++){
        $objeto -> Operaciones($i,$j);
           }
}

$objeto-> imprime();

?>
```

Ver ejemplo75.php

Un ejemplo de clase *extendida*

En este ejemplo puedes ver como las **clases extendidas** utilizan variables de la clase **padre**, pueden tener **constructores propios** pero **solo ejecutan su *propio constructor* pero no el de la clase padre**.

Para que el **constructor** de la **clase padre** sea ejecutado desde la clase extendida tiene que ser invocado expresamente.

```
<?
class Multiplica{

    var $factor1=7;
    var $factor2=8;

    function Multiplica(){
        print $this->factor1*$this->factor2."<br>";
        print "Esto está en el constructor de la clase padre<br>";
    }
}

class MeSeOlvido extends Multiplica{

   var $divisor=5;
   function MeSeOlvido(){
        print $this->factor1*$this->factor2/$this->divisor."<br>";
   }
}

$objeto= new MeSeOlvido;

$objeto->Multiplica();
 ?>
```

Ver ejemplo76.php

Ejemplo de funciones PHP con clases y objetos

En este ejemplo puedes comprobar que las clases pueden escribirse en ficheros externos y posteriormente ser incluidas en un *script* PHP mediante la función **include**.

```
<?
include("ejemplo75.php");

$busca="imprima";

if(method_exists ( $objeto, $busca)){
    echo "Existe la función  $busca <br>";
}else{
    echo "No existe la función $busca <br>";
}

$r=get_class_vars ("Operaciones");
foreach ($r as $pepe=>$pepito){
    echo "$pepe -->$pepito<br>";
}

$s=get_class_methods("Operaciones");
foreach($s as $clave){
    echo $clave,"<br>";
}
print_r(get_object_vars($objeto));
?>
```

Ver ejemplo77.php

Ejemplo de utilización de ::

Devuelve **todas las variables** (y sus valores) contenidas en el *objeto* **obj**.

La llamada ::

PHP permite llamar a una función definida en una **clase** sin necesidad de **crear** un objeto.

La sintaxis es la siguiente:

clase **::** *funcion()*

donde **clase** es el nombre de una **clase** y **funcion()** es una función definida dentro de esa clase.

Su funcionalidad es la misma que si escribiéramos:

$*nombre* **= new** *clase*

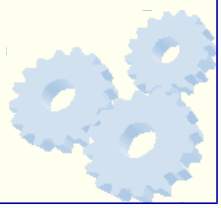
$*nombre* **->** *funcion ()*

```
<?
class A {
    function ejemplo() {
        echo "Este es el resultado de la función ejemplo<br>";
        }
}

# llamo a la funcion sin crear un nuevo objeto
# usando ::
A::ejemplo();

#ahora creo el objeto $b y llamo a la función

$b = new A;
$b->ejemplo();
?>
```



ejemplo78.php