Repaso POO con PHP

Desarrollo de Sistemas Informáticos Web 1

Elaborado por:
Ing. Tomás Eduardo Urbina

La POO es un **paradigma de programación** (o técnica de programación) que utiliza objetos e interacciones en el diseño de un sistema.

La POO está compuesta por una serie de elementos que se detallan a continuación.

Clase

Una clase es un **modelo** que se utiliza para crear objetos que comparten un mismo comportamiento, estado e identidad.

Metáfora

Persona es la metáfora de una clase (la abstracción de Juan, Pedro, Ana y María), cuyo comportamiento puede ser caminar, correr, estudiar, leer, etc. Puede estar en estado despierto, dormido, etc. Sus características (propiedades) pueden ser el color de ojos, color de pelo, su estado civil, etc.

```
class Persona {
# Propiedades
# Métodos
}
```

Objeto

Es una **entidad** provista de métodos o mensajes a los cuales responde (comportamiento); atributos con valores concretos (estado); y propiedades (identidad).

```
$persona = new Persona();
/*
El objeto, ahora, es $persona,
que se ha creado siguiendo el modelo de la clase Persona
*/
```

Método

Es el **algoritmo** asociado a un objeto que indica la capacidad de lo que éste puede hacer.

```
function caminar() {
#...
}
```

Evento y Mensaje

Un **evento** es un suceso en el sistema mientras que un **mensaje** es la comunicación del suceso dirigida al objeto.

Propiedades y atributos

Las propiedades y atributos, son **variables** que contienen datos asociados a un objeto.

```
$nombre = 'Juan';
$edad = '25 años';
$altura = '1,75 mts';
```

La POO debe guardar ciertas características que la identifican y diferencian de otros paradigmas de programación. Dichas características se describen a continuación.

Abstracción

Aislación de un elemento de su contexto. Define las características esenciales de un objeto.

Encapsulamiento

Reúne al mismo nivel de abstracción, a todos los elementos que puedan considerarse pertenecientes a una misma entidad.

Modularidad

Característica que permite dividir una aplicación en varias partes más pequeñas (denominadas módulos), independientes unas de otras.

Ocultación (aislamiento)

Los objetos están aislados del exterior, protegiendo a sus propiedades para no ser modificadas por aquellos que no tengan derecho a acceder a las mismas.

Polimorfismo

Es la capacidad que da a diferentes objetos, la posibilidad de contar con métodos, propiedades y atributos de igual nombre, sin que los de un objeto interfieran con el de otro.

Herencia

Es la relación existente entre dos o más clases, donde una es la principal (madre) y otras son secundarias y dependen (heredan) de ellas (clases “hijas”), donde a la vez, los objetos heredan las características de los objetos de los cuales heredan.

Recolección de basura

Es la técnica que consiste en destruir aquellos objetos cuando ya no son necesarios, liberándolos de la memoria.

Declaración de una clase y creación de un objeto.

La programación orientada a objetos se basa en la programación de clases; a diferencia de la programación estructurada, que está centrada en las funciones.

Una clase es un molde del que luego se pueden crear múltiples objetos, con similares características.

Un poco más abajo se define una clase Persona y luego se crean dos objetos de dicha clase.

Una clase es una plantilla (molde), que define atributos (lo que conocemos como variables) y métodos (lo que conocemos como funciones).

La clase define los atributos y métodos comunes a los objetos de ese tipo, pero luego, cada objeto tendrá sus propios valores y compartirán las mismas funciones.

Debemos crear una clase antes de poder crear objetos (instancias) de esa clase. Al crear un objeto de una clase, se dice que se crea una instancia de la clase o un objeto propiamente dicho.

Confeccionaremos nuestra primera clase para conocer la sintaxis en el lenguaje PHP, luego definiremos dos objetos de dicha clase.

Implementaremos una clase llamada Persona que tendrá como atributo (variable) su nombre y dos métodos (funciones), uno de dichos métodos inicializará el atributo nombre y el siguiente método mostrará en la página el contenido del mismo.

pagina1.php

```
<html>
<head>
<title>Pruebas</title>
</head>
<body>
<?php
class Persona {
  private $nombre;
  public function inicializar($nom)
  {
    $this->nombre=$nom;
  }
  public function imprimir()
  {
    echo $this->nombre;
    echo '<br>';
  }
}

$per1=new Persona();
$per1->inicializar('Juan');
```

```
$per1->imprimir();
$per2=new Persona();
$per2->inicializar('Ana');
$per2->imprimir();
?>
</body>
</html>
```

La sintaxis básica para declarar una clase es:

```
class [Nombre de la Clase] {
  [atributos]
  [métodos]
}
```

Siempre conviene buscar un nombre de clase lo más próximo a lo que representa. La palabra clave para declarar la clase es class, seguidamente el nombre de la clase y luego encerramos entre llaves de apertura y cerrado todos sus atributos (variable) y métodos(funciones).

Nuestra clase Persona queda definida entonces:

```
class Persona {
  private $nombre;
  public function inicializar($nom)
  {
    $this->nombre=$nom;
  }
  public function imprimir()
  {
    echo $this->nombre;
    echo '<br>';
  }
}
```

Los atributos normalmente son privados (private), ya veremos que esto significa que no podemos acceder al mismo desde fuera de la clase. Luego para definir los métodos se utiliza la misma sintaxis que las funciones del lenguaje PHP.

Decíamos que una clase es un molde que nos permite definir objetos. Ahora veamos cual es la sintaxis para la definición de objetos de la clase Persona:

```
$per1=new Persona();
$per1->inicializar('Juan');
$per1->imprimir();
```

Definimos un objeto llamado $per1 y lo creamos asignándole lo que devuelve el operador new. Siempre que queremos crear un objeto de una clase utilizamos la sintaxis new [Nombre de la Clase].

Luego para llamar a los métodos debemos anteceder el nombre del objeto el operador -> y por último el nombre del método. Para poder llamar al método, éste debe ser definido público (con la palabra clave public). En el caso que tenga parámetros se los enviamos:

```
$per1->inicializar('Juan');
```

También podemos ver que podemos definir tantos objetos de la clase Persona como sean necesarios para nuestro algoritmo:

```
$per2=new Persona();
$per2->inicializar('Ana');
$per2->imprimir();
```

Esto nos da una idea que si en una página WEB tenemos 2 menúes, seguramente definiremos una clase Menu y luego crearemos dos objetos de dicha clase.

Esto es una de las ventajas fundamentales de la Programación Orientada a Objetos (POO), es decir reutilización de código (gracias a que está encapsulada en clases) es muy sencilla.

Lo último a tener en cuenta en cuanto a la sintaxis de este primer problema es que cuando accedemos a los atributos dentro de los métodos debemos utilizar los operadores $this-> (this y ->):

```
  public function inicializar($nom)
  {
    $this->nombre=$nom;
  }
```

El atributo $nombre solo puede ser accedido por los métodos de la clase Persona.

Atributos de una clase.

Ahora trataremos de concentrarnos en los atributos de una clase. Los atributos son las características, cualidades, propiedades distintivas de cada clase. Contienen información sobre el objeto. Determinan la apariencia, estado y demás particularidades de la clase. Varios objetos de una misma clase tendrán los mismos atributos pero con valores diferentes.

Cuando creamos un objeto de una clase determinada, los atributos declarados por la clase son localizadas en memoria y pueden ser modificados mediante los métodos.

Lo más conveniente es que los atributos sean privados para que solo los métodos de la clase puedan modificarlos.

Plantearemos un nuevo problema para analizar detenidamente la definición, sintaxis y acceso a los atributos.

Ejercicio: Implementar una clase que muestre una lista de hipervínculos en forma horizontal (básicamente un menú de opciones)

Lo primero que debemos pensar es que valores almacenará la clase, en este caso debemos cargar una lista de direcciones web y los títulos de los enlaces. Podemos definir dos vectores paralelos que almacenen las direcciones y los títulos respectivamente.

Definiremos dos métodos: cargarOpcion y mostrar.

pagina1.php

```
<html>
<head>
<title>Pruebas</title>
</head>
<body>
<?php
class Menu {
  private $enlaces=array();
  private $titulos=array();
  public function cargarOpcion($en,$tit)
  {
    $this->enlaces[]=$en;
    $this->titulos[]=$tit;
  }
  public function mostrar()
  {
    for($f=0;$f<count($this->enlaces);$f++)
    {
      echo '<a href="'.$this->enlaces[$f].'">'.$this->titulos[$f].'</a>';
      echo "-";
    }
  }
```

```
}

$menu1=new Menu();
$menu1->cargarOpcion('http://www.google.com','Google');
$menu1->cargarOpcion('http://www.yahoo.com','Yhahoo');
$menu1->cargarOpcion('http://www.msn.com','MSN');
$menu1->mostrar();
?>
</body>
</html>
```

Analicemos ahora la solución al problema planteado, como podemos ver normalmente los atributos de la clase se definen inmediatamente después que declaramos la clase:

```
class Menu {
  private $enlaces=array();
  private $titulos=array();
```

Si queremos podemos hacer un comentario indicando el objetivo de cada atributo.

Luego tenemos el primer método que añade a los vectores los datos que llegan como parámetro:

```
  public function cargarOpcion($en,$tit)
  {
    $this->enlaces[]=$en;
    $this->titulos[]=$tit;
  }
```

Conviene darle distinto nombre a los parámetros y los atributos (por lo menos inicialmente para no confundirlos).

Utilizamos la característica de PHP que un vector puede ir creciendo solo con asignarle el nuevo valor. El dato después de esta asignación $this->enlaces[]=$en; se almacena al final del vector.

Este método será llamado tantas veces como opciones tenga el menú.

El siguiente método tiene por objetivo mostrar el menú propiamente dicho:

```
  public function mostrar()
  {
    for($f=0;$f<count($this->enlaces);$f++)
    {
      echo '<a href="'.$this->enlaces[$f].'">'.$this->titulos[$f].'</a>';
      echo " | ";
    }
  }
```

Disponemos un for y hacemos que se repita tantas veces como elementos tenga el vector $enlaces (es lo mismo preguntar a uno u otro cuantos elementos tienen ya que siempre

tendrán la misma cantidad). Para obtener la cantidad de elementos del vector utilizamos la función count.

Dentro del for imprimimos en la página el hipervínculo:

```
echo '<a href="'.$this->enlaces[$f].'">'.$this->titulos[$f].'</a>';
```

Hay que acostumbrarse que cuando accedemos a los atributos de la clase se le antecede el operador $this-> y seguidamente el nombre del atributo propiamente dicho. Si no hacemos esto estaremos creando una variable local y el algoritmo fallará.

Por último para hacer uso de esta clase Menu debemos crear un objeto de dicha clase (lo que en programación estructurada es definir una variable):

```
$menu1=new Menu();
$menu1->cargarOpcion('http://www.google.com','Google');
$menu1->cargarOpcion('http://www.yahoo.com','Yhahoo');
$menu1->cargarOpcion('http://www.msn.com','MSN');
$menu1->cargarOpcion('http://www.utec.edu.sv','UTEC');
$menu1->mostrar();
```

Creamos un objeto mediante el operador new y seguido del nombre de la clase. Luego llamamos al método cargarOpcion tantas veces como opciones necesitemos para nuestro menú (recordar que SOLO podemos llamar a los métodos de la clase si definimos un objeto de la misma)

Finalmente llamamos al método mostrar que imprime en la página nuestro menú.