Programación web con PHP

Programación orientada a objetos: Introducción

Temario

Qué es la programación orientada a objetos?

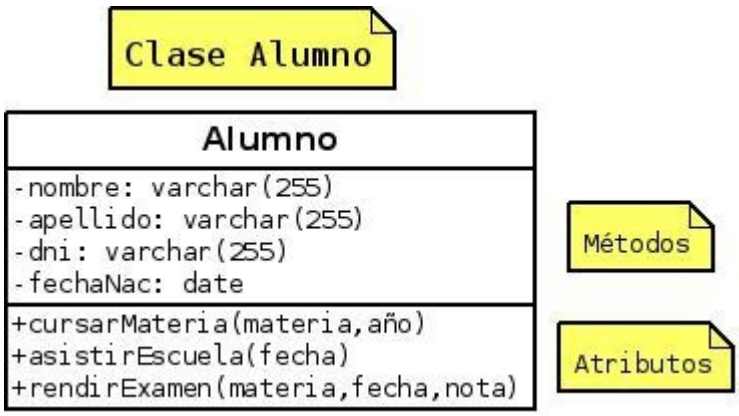
La programación orientada a objetos es un paradigma de programación donde se utilizan objetos y clases. Desde la versión 5 de PHP se ha reescrito y mejorado las características para el manejo de objetos.

En la programación estructurada los algoritmo se ejecutan en orden secuencial mientras que el la programación orientada a objetos contamos con elementos que se relacionan y comunican con otros.

El propósito principal de este paradigma es que los objetos representen de manera menos abstracta los objetos de la vida real.

En una escuela por ejemplo tenemos alumnos, cada uno de estos alumnos tiene ciertas características: nombre, apellido, D.N.I., fecha de Nacimiento,etc y también realiza ciertas acciones: cursar una materia, rendir un examen, asistir a la escuela, etc. Cada uno de los alumnos tiene las mismas características solo que con diferentes valores y realiza las mismas acciones solo que de diferente manera.

Por lo tanto decimos que cada alumno con sus valores de características y acciones que realiza es un “objeto” y al modelo abstracto de características y acciones lo llamamos clase. Sus características se llaman atributos y sus acciones métodos.



Objeto

Un objeto es una unidad de memoria que posee estado y comportamiento.

Siguiendo con el ejemplo anterior un alumno en particular tendrá como nombre “Pedro”, como apellido “Gonzales”, D.N.I 34545665 y fecha de nacimiento 13/12/1989.

Este es el estado del objeto.

Este alumno también podrá o no cursar alguna materia o asistir un día determinado, esto determina su comportamiento.

Clase

Una clase es una plantilla o modelo desde donde es posible crear instancias. Cada uno de los alumnos aunque tomen diferentes valores tendrán nombre, apellido, D.N.I y fecha de nacimiento y cursaran materias y rendirán exámenes aunque no de la misma manera que ellos otros alumnos. Estas características y acciones que tienen en común conforman la clase.

Visibilidad

Los atributos y las funciones permiten restringir su acceso, existen tres configuraciones

- public: el atributo o método puede ser accedido desde afuera del ámbito de la clase ya sea por código exterior o desde otra clase.
- private: el atributo o método solo puede ser accedido desde adentro del ámbito de la clase.
- protected: el atributo o método solo puede ser accedido desde adentro del ámbito de la clase y de las clases que hereden de ella

Creación de clases e instanciación

Para definir una clase colocamos la palabra class seguida del nombre de la clase, luego entre corchetes definimos las variables y funciones pertenecientes a la clase.

Para crear un objeto creamos una instancia de esta clase utilizando la palabra new.

```php
<?php
//Definición clase
class NombreClase
{

}
?>
```

```php
<?php
//Crear instancia
$instancia = new NombreClase;
?>
```

Definir atributos

Las atributos son las variables propias de una clase, para definir los atributos utilizamos las claves public, private y protected seguida por la definición de la variables.

Un atributo también puede ser estático, esto significa que no es necesario instanciar la clase para acceder a el y que su valor no puede ser modificado. Para definir un atributo como estático utilizamos la palabra static.

```php
<?php
class NombreClase
{
//definicion de atributos
 public $edad =32;
 private $nombre =”Pablo”;
 protected $apellido =”Gonzales”;
 public static $tipoSangre =”A+”;

}
?>
```

Acceder a un atributo

Para acceder a un atributo dentro del ámbito de la clase utilizamos $this-> seguido por el nombre del atributo.

Para acceder a un atributo estático utilizamos self:: o parent:: seguido por $ y el nombre del atributo. El primer caso lo utilizamos cuando se trata de un atributo de la misma clase y el segundo cuando es de una clase heredada.

Para acceder a un atributo desde afuera del ámbito de la clase utilizamos el objeto previa

instanciación → y el nombre del atributo. Solo los atributos públicos podrán ser accedidos desde afuera del ámbito de la clase.

Para acceder a un atributo estático utilizamos el nombre de la clase seguido por ::$ y el nombre del atributo.

```php
<?php
//Crear instancia
$instancia = new NombreClase;
echo $instancia>edad;
echo $instancia::$tipoSangre;
?>
```

Definir métodos

Los métodos son las funciones propias de una clase, su funcionamiento y sintaxis es similar a las funciones de PHP. De la misma manera que los atributos puede ser definida como public, private o protected.

Existen un tipo especial de métodos llamados métodos abstractos que no tienen código, simplemente se define el método ya que su funcionalidad será dada en una herencia.

```php
<?php
class NombreClase
{
        public $nombre="Maria";
        public $apellido="Moreno";

       //Defino un método
        public function mostrarMensaje()
        {
                return "Mi nombre es ".$this>nombre." ".$this>apellido;
        }
}
?>
```

Acceder a un método

Para acceder a un atributo dentro de un método de la misma clase utilizamos $this-> y el nombre del método

Para acceder a un método después de instanciar la clase utilizamos → y el nombre del método.

```php
<?php
$instancia= new NombreClase;
echo $instancia>mostrarMensaje();
?>
```

Constructores

Un constructor es un método especial que permiten inicializar las propiedades de un objeto, al instanciarse automáticamente se ejecuto el constructor.

Para definir un constructor utilizamos un doble guión bajo.

```php
<?php
class NombreClase
{
        public $nombre="Maria";
        public $apellido="Moreno";

        //Defino el constructor
        function __construct($nombre,$apellido)
         {
                //seteo los atributos
                 $this>nombre = $nombre;
                 $this>apellido = $apellido;
         }
}
//instancio pasando los parámetros para el constructor
$instancia= new NombreClase(“Pedro”,”Gonzales”);

?>
```

Destructores

Un destructor es una función especial que es llamada cuando un objeto es destruido.

Los destructores pueden ser llamados en dos situaciones, al terminar el script o al destruir el objeto con unset():

El destructor se define con dos guiones bajos.

```
<?php
class NombreClase
{
//Defino el destructor
 function __destruct()
 {
        echo "Destruyendo<br>";
 }

}
?>
```

Encapsulamiento

El encapsulamiento es un mecanismo que se utiliza para evitar que los atributos de una clase sean solo modificados y accedidos a través de los métodos de una clase. De esta manera se garantiza la integridad de los datos.

Para implementarlo definimos todos los atributos como privados y creamos dos métodos públicos para cada uno, uno que permitirá el ingreso del dato, y uno que devolverá el valor actual.

```
<?php

class Clase
{
        private $nombre;
        private $apellido;

        public function getNombre()
        {
                return $this>nombre;
        }
        public function setNombre($value)
        {
                $this>nombre=$value;
        }

        public function getApellido()
```

```
	{
		return $this>apellido;
	}
	public function setApellido($value)
	{
		$this>apellido=$value;
	}

}

$instancia=new Clase();
$instancia>setNombre("Maria");
$instancia>setApellido("Moreno");
echo $instancia>getNombre();//Maria
echo $instancia>getApellido();//Moreno

?>
```