Clases y Objetos en PHP

PHP es un lenguaje que soporta la programación orientada a objetos, para definir una clase en PHP, se realiza mediante la palabra reservada *class*, seguido del nombre que se le dará a la clase, luego se abre y se cierran llaves, allí se contendrá todos los atributos o propiedades, métodos de la clase.

¿Qué es una clase?

Una clase es una plantilla que define las propiedades y métodos que tendrán los objetos. Se declara con la palabra clave class.

```php
<?php
class Automovil
{
    // Declaración de propiedades
    public string $modelo;
    protected string $placa;
    private int $anio;

    // Constructor
    public function __construct(string $modelo = "", string $placa = "", int $anio = 0)
    {
        $this->modelo = $modelo;
        $this->placa = $placa;
        $this->anio = $anio;
    }

    // Método
    public function imprimirPropiedades(): void
    {
        echo "El modelo es: {$this->modelo}";
        echo " | Placa: {$this->placa}";
        echo " | Año: {$this->anio}";
    }
}
?>
```

Crear el objeto

Para crear o instanciar un objeto de tipo *Automovil* se hace de la siguiente manera:

1. Nombre del objeto
2. Nombre de la clase de la cual se quiere crear el objeto

Veamos:

```php
$objAutomovil = new Automovil("Civic", "XYZ123", 2021);
```

Acceder a las propiedades de la clase:

```php
$objAutomovil = new Automovil("Civic", "XYZ123", 2021);
echo $objAutomovil->modelo;
```

Ejemplo Completo:

```php
<?php
class Automovil
{
    // Declaración de propiedades
    public string $modelo;
    protected string $placa;
    private int $anio;

    // Constructor
    public function __construct(string $modelo = "", string $placa = "", int $anio = 0)
    {
        $this->modelo = $modelo;
        $this->placa = $placa;
        $this->anio = $anio;
    }

    // Método
    public function imprimirPropiedades(): void
    {
        echo "El modelo es: {$this->modelo}";
        echo " | Placa: {$this->placa}";
        echo " | Año: {$this->anio}";
    }
}

$objAutomovil = new Automovil();
echo $objAutomovil->modelo; // Accede a propiedad pública
$objAutomovil->imprimirPropiedades();
?>
```

Recordar:

- public: accesible desde cualquier parte.
- protected: accesible desde la clase y sus descendientes.
- private: accesible solo dentro de la clase

Propiedades tipadas y promoción de constructor

En la versión 7.4 se introdujo propiedades tipadas:

```php
class Automovil
{
    // Declaración de propiedades
    public string $modelo;
    protected string $placa;
    private int $anio;
```

En la versión 8 permite simplificar el constructor:

```php
<?php
class Automovil
{
    public function __construct(
        public string $modelo = "Accord",
        protected string $placa = "XYZ123",
        private int $anio = 2019
    ) {}

    public function imprimirPropiedades(): void
    {
        echo "El modelo es: {$this->modelo}";
        echo " | Placa: {$this->placa}";
        echo " | Año: {$this->anio}";
    }
}

$obj = new Automovil("Civic", "ABC456", 2022);
$obj->imprimirPropiedades();
?>
```

¿Son necesarios los destructores en PHP?

Realmente ya no son necesarios en la mayoría de los casos. PHP actualmente gestiona en forma automática la memoria y los recursos cuando el script termina o cuando el objeto ya no se usa. Esto significa que cuando tu script finaliza, PHP destruye todos los objetos automáticamente, liberando la memoria.

Sin embargo, si se usan cuando se tiene que: cerrar recursos manuales (archivos abiertos, conexiones a sockets, etc.), liberar bloqueos sobre recursos externos, registrar logs.

Autocarga de clases

Para evitar require de cada archivo, puedes usar spl_autoload_register().

Ejemplo:

```php
<?php
function autoCarga($class)
{
    require_once "clases/{$class}.php";
}
spl_autoload_register('autoCarga');
?>
```

Con función anónima:

```php
<?php
spl_autoload_register(function($class) {
    require_once "clases/{$class}.php";
});
?>
```

Cuando el proyecto crece, es mejor usar Composer con PSR-4:

```
composer init
composer dump-autoload
```