

DESARROLLO WEB EN ENTORNO SERVIDOR

CAPÍTULO 8: Programación con código embebido en el servidor Clases.

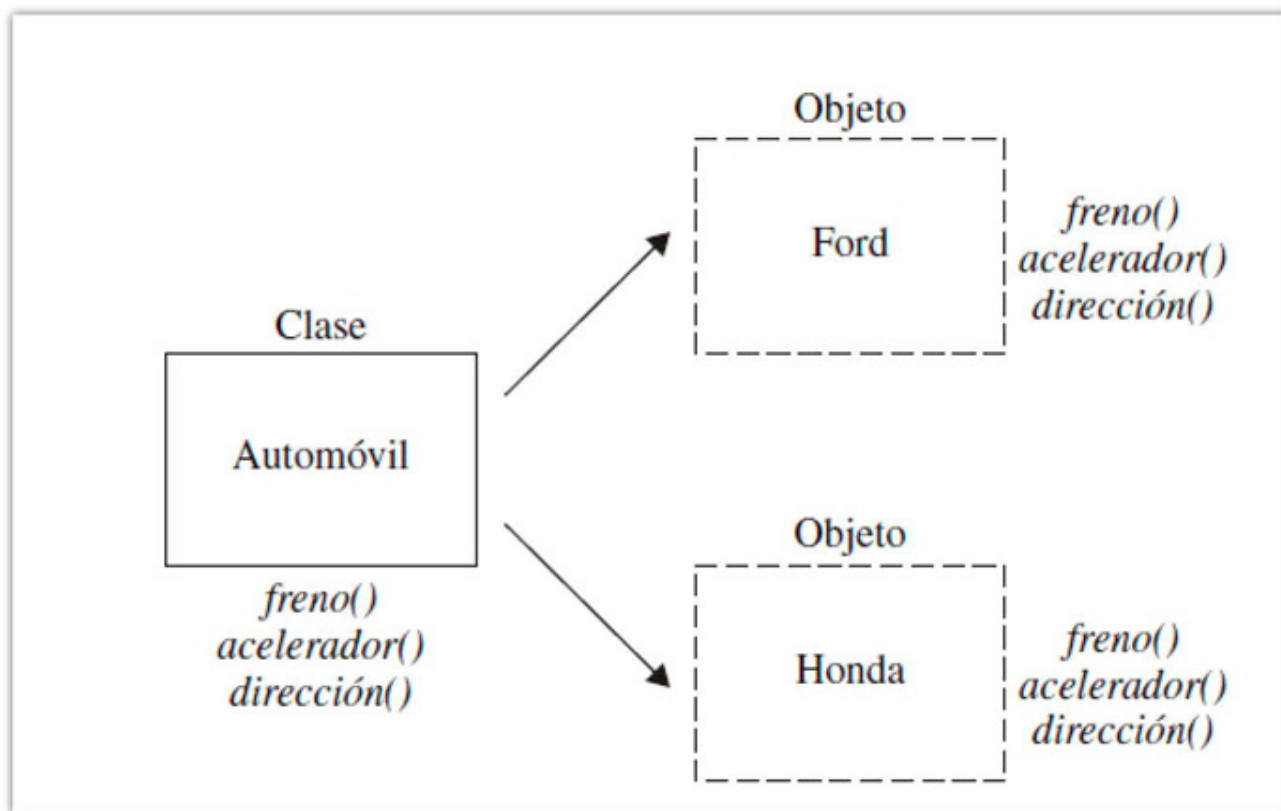
Introducción

- Una clase permite agrupar funciones relacionadas.
- Son la base en la programación orientada a objetos (POO)
- Las clases son plantillas que permiten crear copias funcionales (instancias) y cada copia es un objeto de la clase.
- Los objetos son independientes entre sí pero sus posibilidades generales dependen de la clase.

Estructura

- Una clase (coches) es una colección una colección autosuficiente e independiente de variables y funciones que trabajan juntas para realizar una o más tareas específicas
 - Las **variables** dentro de las clases reciben el nombre de propiedades o atributos (color, acabado...)
 - Las **funciones** dentro de las clases se llaman métodos (acelerar, frenar, girar...)
- Cada copia o instancia de la clase (seat, ford...) crea un objeto con las propiedades y métodos de la clase pero pueden **parametrizarse** de forma independiente.

Relación entre clases y objetos



Clases-Definición-Ejemplo

```
<?php
class Automovil{                                // define clase
    public $color;                               // define propiedades
    public $modelo;

                                                // define métodos
    public function acelerador(){
        echo 'Aumenta la velocidad...';
    }
    public function freno(){
        echo 'Disminuye la velocidad...';
    }
    public function direccion(){
        echo 'Da una vuelta...';
    }
}
?>
```


Clases-Definición

- Una definición de clase comienza con la palabra clave **class**, seguida por el nombre de la clase y un par de llaves({ }).
- Dentro de la clase se definen, casi siempre, **variables** (propiedades) y las **funciones** (métodos)
- Las propiedades y métodos tienen 3 niveles de visibilidad, de mayor a menor y que son **public**, **protected** y **private**
 - El nivel de visibilidad otorgado establecerá que una función pueda manipular o no los valores desde el cuerpo principal del programa.
 - El valor por defecto es public (si no se indica, queda establecido por defecto)

Objetos-Definición-Ejemplo

```
<?php
// crea objeto coche

$coche= new Automovil;
// establece propiedades

$coche->color = 'rojo';
$coche->modelo = 'Ford ';
// invoca métodos del objeto

$coche->acelerador();
$coche->direccion();
?>
```

- La palabra clave para crear (instanciar un nuevo objeto), es **new** seguida del nombre de la clase.
- Los objetos pueden acceder de manera directa a los métodos y las propiedades de la clase.
- El símbolo **->** se emplea para conectar objetos a sus propiedades o métodos, se llama **operador de alcance o de objeto**

Acceso a los métodos desde la clase

```
<?php
class Automovil {
    public $color;
    public $modelo;
    public $velocidad = 55;

    public function acelerador(){
        $this->velocidad += 10;
        echo 'Aumenta la velocidad a ' . $this->velocidad .
            '...';
    }
    public function freno(){
        $this->velocidad -= 10;
        echo 'Disminuye la velocidad a ' . $this->velocidad .
            '...';
    }
    public function direccion(){
        $this->freno();
        echo 'Da una vuelta...';
        $this->acelerador();
    }
}
?>
```

- Para acceder o modificar un **método** o **propiedad** de clase desde la misma clase es necesario añadir un prefijo al método o la propiedad correspondiente; dicho prefijo es **\$this**, que hace referencia a “esta” clase.
- En el ejemplo se establece una propiedad de clase llamada **\$velocidad** y luego se modifica desde las funciones **acelerador()** y **freno()**, posteriormente se llama a ambos métodos desde **direccion**.

Acceso a los métodos desde los objetos

```
<?php
// creamos objeto
$coche= new Automovil;
// invoca métodos

$coche->acelerador();
$coche->direccion();

// datos de salida:
// 'Acelerador a 65...'
// 'Freno a 55...'
// 'Da una vuelta...'
// 'Acelerador a 65...'

?>
```

- El símbolo **->** se emplea para invocar a los métodos de la clase de la que depende el objeto
- Para acceder o modificar un método o propiedad de la clase desde el objeto se emplea el mismo símbolo **->** pero en éste caso en lugar de la variable **\$this**, lo aplicamos sobre la variable que contiene el nuevo objeto, **\$coche** para el ejemplo.

Inspeccionar Clases y objetos

- Desde la versión 5.0, PHP te permite *inspeccionar* cualquier clase u objeto para obtener información detallada sobre sus características
- Para utilizarlo, crea un objeto **ReflectionClass** o **ReflectionObject** y aplícalo a la clase u objeto a inspeccionar.

```
<?php
```

```
// reflejo para clase
```

```
    Reflection::export(new ReflectionClass('Automovil'));
```

```
// reflejo para objeto
```

```
    Reflection::export(new ReflectionObject('$coche'));
```

```
?>
```

Ejercicio 1 - Clave

- Realiza el ejercicio de la página 139 de tu manual de referencia (PDF)
- Nota:
 - En lugar de la clase Jumbler, llámala **codificador**
 - Guarda la página como **ClaseCodificadora.php**
- **NOTA**
 - Lo más normal es que las clases estén definidas en su propio fichero, generalmente en una carpeta "class" y dicho fichero es incluido (include), en las páginas que se requiera la construcción de objetos dependientes

Ejercicio 2 – Clase persona

- Crear una clase llamada **Persona**.
- Definir una propiedad donde se almacene su nombre.
- Luego definir dos métodos, uno que cargue el nombre y otro que lo imprima.
- Crear dos objetos nuevos basados en la clase

Guarda el fichero como **ClasePersona.php**

Ejercicio 3 – Clase empleado

- Crear una clase llamada **empleado**.
- Definir como propiedades nombre y sueldo.
- Definir un método de inicio al que le llegue como datos nombre y sueldo.
- Crear un segundo método que imprima el nombre y si debe o no pagar impuestos si el sueldo es mayor a 3000€
- Crear un objeto basado en la clase

Guarda el fichero como **ClaseEmpleado.php**

Constructores y Destruktores

- Son funciones, o métodos, que se encargan de ejecutar código automáticamente cuando se instancian los objetos.
- Para ello disponemos de dos métodos de clase especiales:
 - El *constructor*, ejecuta código al instanciar el objeto.
 - *destructor* ejecuta código cuando el objeto termina su acción
- Se implementan definiendo las funciones llamadas `__construct()` y `__destruct()` dentro de la clase y colocando el código, según el caso, dentro de su respectivo método.
- Ambos pertenecen al grupo "métodos mágicos en php"
<http://php.net/manual/es/language.oop5.magic.php>

Ejemplo Constructor/Destructor

```
<?php
    class Maquina {                                // define clase

        function __construct(){                    // constructor
            echo "Encendido...\n";
        }

        function __destruct(){                      // destructor
            echo "Apagado...\n";
        }
    }

    $m = new Maquina(); // crea un objeto, dato de salida: "Encendido..."
    unset ($m); // luego lo destruye dato de salida: "Apagado..."
?>
```

Ejercicio 4 – Constructor

- Crea una clase **CabeceraPagina** que permita mostrar un título, indicarle si queremos que aparezca centrada, a derecha o izquierda.
- Utilizar un constructor para inicializar los dos atributos.
- NO DESTRUIR

Ejercicio 4 – Constructor: Solución

```
<?php //Clase
class CabeceraPagina {
    private $titulo;
    private $ubicacion;
    public function __construct($tit,$ubi) {
        $this->titulo=$tit;
        $this->ubicacion=$ubi;
    }
    public function dibuja() {
        echo '<div style="font-size:40px;text-align:'.$this->ubicacion.'">';
        echo $this->titulo;
        echo '</div>';
    }
}
?>
```

```
<?php //objeto
$cabecera=new
CabeceraPagina('Aprendiendo
programación','center');
$cabecera->dibuja();
?>
```

Herencia o extensibilidad

- *La Herencia significa que una nueva clase puede estar basada en una clase existente (hija o subclase), heredando todas las propiedades y métodos de ésta. (Clase Padre o superclase)*
- Las nuevas clases son versiones "extendidas" de la principal
- Se puede añadir nuevas propiedades y métodos conforme se vayan necesitando tanto en la clase principal como en las extendidas.
- En PHP una clase sólo puede derivar de una única clase, es decir, PHP no permite herencia múltiple.

Herencia o extensibilidad

- La estructura esencial para definir una clase heredada (o hija) es emplear la palabra clave **extends** seguida del nombre de la clase Padre

```
<?php
    class Mamiferos {                                // define clase
    }
    class Humano extends Mamiferos { // crea subclase
    }
?>
```

Ejercicio 5 – Herencia

- Confeccionar una clase llamada **Operacion** que defina como propiedades **\$valor1**, **\$valor2**, **\$resultado** y defina como métodos **cargar1** (inicializa el atributo **\$valor1**), **cargar2** (inicializa el atributo **\$valor2**) y por último un método que muestre el contenido de **\$resultado**.
- Luego definir **dos subclases** de la clase **Operacion**. La primera llamada **Suma** que tiene por objetivo la carga de dos valores, sumarlos y mostrar el resultado. La segunda llamada **Resta** que tiene por objetivo la carga de dos valores, restarlos y mostrar el resultado de la diferencia.

Ejercicio 5 – Herencia: Solución (I)

```
<?php
```

```
class Operacion {  
    protected $valor1;  
    protected $valor2;  
    protected $resultado;  
    public function cargar1($v)  
    {  
        $this->valor1=$v;  
    }  
    public function cargar2($v)  
    {  
        $this->valor2=$v;  
    }  
    public function imprimirResultado()  
    {  
        echo $this->resultado.'<br>';  
    }  
}
```

```
//subclases
```

```
class Suma extends Operacion{  
    public function operar()  
    {  
        $this->resultado=$this->valor1+$this->valor2;  
    }  
}  
class Resta extends Operacion{  
    public function operar()  
    {  
        $this->resultado=$this->valor1-$this->valor2;  
    }  
}
```

Ejercicio 5 – Herencia: Solución (II)

//Creamos los objetos

```
$suma=new Suma();  
    $suma->cargar1(10);  
    $suma->cargar2(10);  
    $suma->operar();  
echo 'El resultado de la suma de 10+10 es:';  
$suma->imprimirResultado();  
$resta=new Resta();  
$resta->cargar1(10);  
$resta->cargar2(5);  
$resta->operar();  
echo 'El resultado de la diferencia de 10-5 es:';  
$resta->imprimirResultado();
```

?>

Herencia y Polimorfismo

- **Polimorfismo:** cualidad por la cual podemos redefinir las propiedades y métodos que una subclase ha heredado, de manera que el cambio afecte sólo a esa subclase.
- Más referencias sobre al herencia y el polimorfismo aquí:
 - http://phpbasico.freevar.com/temas/php15_4.php

Visibilidad

- Para ejercer más control sobre los objetos tenemos 3 niveles de visibilidad para propiedades y métodos:
 - **Public** (x defecto) la función invocadora puede modificarlos directamente desde el cuerpo principal del programa.
 - **Private**: visibilidad sólo para la clase que los define
 - **Protected**: visibilidad para la clase que los define y las subclases (heredadas)

Visibilidad: Ejemplo

```
<?php
// árbol de clases
class Mamiferos {
    public $nombre;
    protected $edad;
    private $especie;
}

class Humano extends Mamiferos {

    $mamifero = new Mamifero;
    $mamifero->nombre = 'William'; // ok
    $mamifero->edad = 3; // error fatal
    $mamifero->especie = 'Ballena'; // error fatal
    $humano = new Humano;
    $humano->nombre = 'Beto'; // ok
    $humano->edad = '1'; // error fatal
    $humano->especie = 'Niño'; // sin definir
}

?>
```

Ejercicio Completo

- Realiza el ejercicio de la página 148 de tu manual de referencia (PDF)
- NO lo copies y lo pegues sin más, LEE.
- Cuando lo copies presta atención a donde copias y pegas los contenidos.
 - Guarda la página cómo [SelectDinamicos.php](#)

