

DAW2- Desarrollo Web en Entorno Servidor



UD-3 Lenguaje PHP

Sesión 3

Repaso sesión 2



- Arrays
- Funciones
 - Argumentos
 - Devolver valores
 - Funciones variables
- Inclusión de ficheros
- Manejo de ficheros

Índice



- Clases y objetos
 - Modificadores de acceso
 - Declaración de tipos
 - Métodos estáticos
 - Operador de resolución ::
- Herencia
- Abstracción
- Interfaces

Clases y objetos



- **Clase y Objeto** son los conceptos básicos de la *Programación Orientada a Objetos* que giran en torno a las entidades de la vida real. Una clase es un modelo o prototipo definido por el usuario a partir del cual se crean los objetos. Básicamente, una clase combina los campos y métodos (función miembro que define acciones) en una sola unidad.
- En PHP, las clases admiten polimorfismo, herencia y también proporcionan el concepto de clases derivadas y clases base.

Clases y objetos

- Para declarar una clase en PHP se utiliza la palabra clave ***class***, seguida de un nombre de clase, y continuando con un par de llaves que encierran las definiciones de las propiedades y métodos pertenecientes a dicha clase.
- Un **nombre válido** de clase comienza con una letra o un guión bajo, seguido de una cantidad arbitraria de letras, números o guiones bajos.
- Por convención vamos a escribir la primera letra en mayúsculas y sólo una clase por archivo *php*.
- Utilizaremos las funciones `include` o `require` para importar las clases y poderlas utilizar.

Clases y objetos



```
class ClaseBasica
{
    // Declaración de una propiedad/atributo
    private $atributo = 'un valor predeterminado';

    // Declaración de un método
    public function mostrarAtributo() {
        echo $this->atributo;
    }
}
```

Clases y objetos



- Los **constructores** de la clase se utilizan para inicializar nuevos objetos (sólo admite uno)
 - `__construct()`
- Los **propiedades/atributos** son variables que proporcionan el estado de la clase y sus objetos, se accede a ellos con ->
 - `$instanciaClase->nombreAtributo`
- Los **métodos** son funciones que se utilizan para implementar el comportamiento de la clase y sus objetos, se accede a ellos con ->
 - `$instanciaClase->nombreMetodo`

Clases y objetos

- La palabra clave ***this*** (seudovariable ***\$this***) en PHP se usa para referirse a la instancia actual de la clase. También se utiliza para diferenciar entre los parámetros del método y los campos de clase si ambos tienen el mismo nombre.

```
class Coche {  
    //...  
  
    public function getColor()  
    {  
        return $this->color;  
    }  
}
```


Clases y objetos: Modificadores de acceso

- Los modificadores de acceso son palabras clave que se usan para especificar la accesibilidad declarada de un atributo o un método de la clase. En PHP hay tres:
 - **public:** El acceso no está restringido.
 - **protected:** El acceso está limitado a la clase contenedora o a los tipos derivados de la clase contenedora.
 - **private:** El acceso está limitado al tipo contenedor.

Clases y objetos

```
<?php
class Coche{

    // Atributos
    private $modelo;
    private $color;
    private $velocidad;

    // Constructor
    public function __construct($modelo, $color, $velocidad = 0){
        $this->modelo = $modelo;
        $this->color = $color;
        $this->velocidad = $velocidad;
    }

    //Métodos
    public function getColor(){
        // Devolvemos el valor del atributo color
        return $this->color;
    }
    public function setColor($color){
        // Asignamos el valor al atributo color
        $this->color = $color;
    }
    // Resto de métodos
}
```

Clases y objetos

- Inicializando un objeto o instanciando un objeto
 - El operador ***new*** crea una instancia de una clase asignando memoria para un nuevo objeto y devolviendo una referencia a esa memoria. El ***new*** también invoca al constructor de la clase.

```
<?php
```

```
include 'Coche.php';
```

```
$instanciaCoche = new Coche('BMV', 'negro');
```

```
echo "Color: ".$instanciaCoche-&gtgetColor();
```

Clases y objetos: Determinación de tipos

- En PHP también es posible utilizar el *type declaration*, o la determinación de tipos, que permite especificar el tipo de datos que se espera de un argumento en la declaración de una función.
- Los tipos soportados por esta característica son: arrays, clases, interfaces, callables, bool, string, int, y float.

```
<?php
```

```
class Aguacate {  
    public $calorias = 100;  
}  
class Persona {  
    public $energia;  
    public function comerAguacate (Aguacate $aguacate) {  
        $this->energia += $aguacate->calorias;  
    }  
    public function mostrarEnergia(){  
        return "Energía actual: " . $this->energia . "<br>";  
    }  
}
```

```
$sonia = new Persona;  
$almuerzo = new Aguacate;  
$sonia->comerAguacate($almuerzo);  
$sonia->comerAguacate($almuerzo);  
echo $sonia->mostrarEnergia(); // Devuelve: Energía actual 200
```

```
$almuerzo = "Kiwi";  
$sonia->comerAguacate($almuerzo); // Devuelve: Catchable fatal error: Argument 1 passed to  
Persona::comerAguacate() must be an instance of Aguacate, string given
```

Clases y objetos: Métodos estáticos

- Declarar propiedades o métodos de clases como estáticos los hacen accesibles sin la necesidad de instanciar la clase.
- Debido a que los métodos estáticos se pueden invocar sin tener creada una instancia del objeto, la pseudovariable *\$this* no está disponible dentro de los métodos declarados como estáticos.
- Cuando queramos acceder a una constante o método estático usamos *self*.

```
<?php
class Foo {
    public static function unMetodoEstatico() {
        // ...
    }
}
```

```
Foo::unMetodoEstatico();
$nombre_clase = 'Foo';
$nombre_clase::unMetodoEstatico();
```

Clases y objetos: Operador de resolución ::

- El operador de resolución de Ámbito o doble dos-puntos permite acceder a elementos estáticos, constantes, y sobrescribir propiedades o métodos de una clase.
- Cuando se hace referencia a estos elementos desde el exterior de la definición de la clase se utiliza el nombre de la clase.

```
<?php
class MyClass {
    const CONST_VALUE = 'Un valor constante';
}


$classname = 'MyClass';
echo $classname::CONST_VALUE; // A partir de PHP 5.3.0

echo MyClass::CONST_VALUE;
```

Herencia



- Para heredar de una clase en PHP se usa la palabra clave *extends*.
- Los métodos de la clase padre pueden ser sobrescritos por la clase hija.
- Cuando queramos acceder a una constante o método de una clase padre, la palabra reservada *parent* nos sirve para llamarla desde una clase extendida.
- PHP no admite herencia múltiple pero se puede simular algo parecido haciendo uso del mecanismo llamado *traits*.



```
<?php
class ParentClass
{
    public function printItem($string)
    {
        echo 'Parent: ' . $string . PHP_EOL;
    }

    public function printPHP()
    {
        echo 'PHP is great.' . PHP_EOL;
    }
}


class ChildClass extends ParentClass
{
    public function printItem($string)
    {
        echo 'Child: ' . $string . PHP_EOL;
    }
}

$parent = new ParentClass();
$child = new ChildClass();
$parent->printItem('Hola'); // Salida: 'Parent: Hola'
$parent->printPHP();       // Salida: 'PHP is great'
$child->printItem('Adiós'); // Salida: Child: Adiós'
$child->printPHP();        // Salida: 'PHP is great'
```

Abstracción



- En PHP, la abstracción se logra con la ayuda de clases abstractas.
 - Una clase abstracta se declara con la palabra clave ***abstract***.
 - En PHP, no se le permite crear objetos de la clase abstracta.
 - La clase que contiene la palabra clave *abstract* con algunos de sus métodos (no todos los métodos abstractos) se conoce como clase base abstracta.
 - La clase que contiene la palabra clave ***abstract*** con todos sus métodos se conoce como clase base abstracta pura.
 - No se permite declarar los métodos abstractos fuera de la clase abstracta.
 - Cuando una clase hereda de una abstracta, si ésta tiene un método abstracto, debe ser definido en la clase hija.



```
<?php
abstract class CocheAbstract {


    public function getRuedas()
    {
        return 4;
    }
    abstract public function setPotencia($potencia);
    abstract public function getPotencia();
}

class Audi extends CocheAbstract {
    public $brand = 'Audi';
    protected $potencia;
    public function setPotencia($potencia)
    {
        $this->potencia = $potencia;
    }
    public function getPotencia()
    {
        return $this->potencia;
    }
}
```

Interfaces



- Una interfaz contiene las definiciones de un grupo de funcionalidades relacionadas que una clase o una estructura no abstracta deben implementar.
- Mediante las interfaces puede incluir, por ejemplo, un comportamiento de varios orígenes en una clase.
- Para definir una interfaz, deberá usar la palabra clave ***interface***, tal y como se muestra en el ejemplo siguiente.
- Para implementar la interfaz se utiliza la palabra clave ***implements***.



```
<?php
interface Automovil {
    public function getTipo();
    public function getRuedas();
}
class Coche implements Automovil {
    public function getTipo(){
        echo "Coche";
    }
    public function getRuedas(){
        echo "4";
    }
}
class Moto implements Automovil {
    public function getTipo(){
        echo "Moto";
    }
    public function getRuedas(){
        echo "2";
    }
}
```

Recursos



- <https://www.tutorialrepublic.com/php-tutorial/php-classes-and-objects.php>
- <https://informaticapc.com/tutorial-php/clases-objetos-herencia.php>
- <https://desarrolloweb.com/articulos/sobrecarga-constructores-php.html>
- <https://diego.com.es/interfaces-en-php>
- <https://diego.com.es/type-hinting-en-php>
- <https://codely.tv/blog/screencasts/tipos-php-7/>