

PHP Orientado a Objetos

Fundamentos

<https://www.php.net/manual/es/language.oop5.php>

- La sintaxis para crear una clase en PHP es bastante sencilla: usando la palabra reservada **class**
- Dentro de una clase podremos declarar **propiedades** y **métodos** muy parecido a como lo hacíais en JAVA
- Para acceder a una propiedad o método de la clase desde cualquier punto de ella será ESCRITAMENTE necesario utilizar **\$this**
- Para acceder a cualquier elemento (método o propiedad) de un objeto, en PHP se utiliza el operador **->** (no el . como en JAVA)

Estructuración de una clase

```
<?php
class persona
{
    public $nombre;
    public $apellido;
    private $fecha_nac;

    public function nombre_completo ()
    {
        $nom = $this->nombre." ".$this->apellido;
        return $nom;
    }
}
?>
```

No necesito poner
\$ delante de la
propiedad, solo
del this

Ejemplo de clase básica

Creación de una clase persona con:

- 2 propiedades públicas
- 1 propiedad privada
- 1 método que devuelve el nombre de la persona

- Para indicar desde dónde y cómo se puede acceder a los atributos y a los métodos utilizaremos las palabras reservadas:

- **public**: podrán ser accedidos desde cualquier parte, incluso desde fuera de la clase.
- **protected**: sólo podrán ser accedidos desde el interior de la propia clase o de sus hijos.
- **private**: sólo podrán ser accedidos desde el interior de la clase donde se han definido.
- **static**: podrán ser accedidos sin necesidad de instanciar la clase.

Ámbito de atributos y métodos

Métodos mágicos

- Un método mágico es un método que ya existe por defecto en cualquier clase que creamos. Pero estos métodos estarán vacíos, deberemos sobrecargarlos siempre.
- Un método mágico suele ser llamado de forma automática, sin que nosotros lo llamemos de forma explícita.
- El ejemplo más sencillo es el método __construct().
- Este método nos permite crear un objeto nuevo de una clase, pero nunca lo llamamos de forma explícita si no que se llama automáticamente cuando hacemos un **new**

Métodos mágicos

- No podremos crear ningún método con el mismo nombre de un método mágico.
- Habrá que **sobrecargarlos** para que hagan lo que queramos.
- Siempre van precedidos de 2 guiones bajos (__)
- `__construct()`: método constructor

Métodos mágicos

```
public function __construct ($n, $a, $f)
{
    $this->nombre = $n;
    $this->apellido = $a;
    $this->fecha_nac = $f;
}
```



- **__destruct()**: Permite destruir el objeto y liberar su memoria.
- **__get()**: Se utiliza para consultar el contenido de las propiedades privadas de la clase.
- **__set()**: Se utiliza para introducir contenido en las propiedades privadas de la clase
- **__isset()**: Se utiliza para ver si tiene contenido una propiedad definida como privada
- **__unset()**: Se utiliza para vaciar una propiedad privada.

Métodos mágicos


```
<?php
class persona
{
    public $nombre;
    public $apellido;
    private $fecha_nac;

    public function __get($propiedad)
    {
        return $this->$propiedad;
    }

    public function __set($propiedad, $valor)
    {
        $this->$propiedad = $valor;
    }
}

$yo = new persona ();

$yo->__set("fecha_nac", "12/05/2018");

echo $yo->__get("fecha_nac");
```

?>

Métodos mágicos


- **__toString()**: método que será llamado cuando intentemos mostrar por pantalla un objeto (completo) de esta clase

Métodos mágicos

```
<?php
class persona
{
    public $nombre;
    public $apellido;
    private $fecha_nac;

    public function __toString()
    {
        $salida = "Esta persona se llama $this->nombre
        $this->apellido y nació el $this->fecha_nac";
        return $salida;
    }
}

$yo = new persona ();
echo $yo;
```



El método `__toString` permite que se pueda mostrar el objeto "yo" como si fuera una variable normal

Más métodos mágicos en:
<http://us2.php.net/manual/es/language.oop5.magic.php>

- Hacer un documento PHP que tenga una clase “animal”. Esta clase tendrá como propiedades: Nombre, color y fecha de nacimiento (cadena de texto con formato YYYY-MM-DD).
 - Se deberá hacer el método set y el método toString de esta clase.
 - Se deberá hacer un método que devuelva la edad del animal.
- Hacer un formulario que pida al usuario nombre, color y fecha de nacimiento de un animal.
- Haciendo uso de la clase anterior, mostrar por pantalla los datos del animal introducido por el usuario.

Ejercicios

Archivo animal.php

```
class animal
{
    public $nombre;
    public $color;
    protected $fecha_nacimiento;
    public function __construct ($n='', $c='', $f='')
    {
        $this->nombre = $n;
        $this->color = $c;
        $this->f_nac = $f;
    }
    public function __set($nombre, $valor)
    {
        $this->$nombre = $valor;
    }
    public function __toString()
    {
        $edad = $this->obtener_edad();
        $cadena = "$this->nombre es de color $this->color
        y tiene $edad años";
        return $cadena;
    }
    public function obtener_edad()
    {
        $fecha = strtotime($this->fecha_nacimiento);
        $segundos = time()-$fecha;
        $edad = round($segundos/60/60/24/365);
        return $edad;
    }
}
```

Formulario.php

```
<body>
<form action='#' method='post'>
    Nombre del animal
    <input type='text' name='nombre'>
    <br>
    Color del animal
    <input type='text' name='color'>
    <br>
    Fecha de nacimiento del animal
    <input type='date' name='fecha'>
    <br>
    <input type='submit' name='enviar'>
</form>
<?php
require_once('./animal.php');
if(isset($_GET['enviar']))
{
    $nom = $_GET['nombre'];
    $col = $_GET['color'];
    $nac = $_GET['fec'];

    $ani = new animal ($nom, $col, $nac);

    $edad = $ani->edad();

    echo "El animal introducido tiene $edad años";
}
?>
```

Si queremos mostrar directamente desde un echo, el resultado de una función lo podemos hacer Usando { }

```
$cadena = "$this->nombre es de color $this->color
y tiene {$this->obtener_edad()} años";
```

- Crear una clase «Vehículo». Esta clase tendrá 3 propiedades: nombre, tipo y peso. Ambas propiedades serán privadas.
 - En la propiedad **tipo** se guardará
 - 'C' para camión
 - 'M' para moto
 - 'T' para turismo
 - En la propiedad **peso** se guardarán las toneladas que pesa el vehículo.
- Además, la clase debe tener los métodos:
 - Constructor: para crear un objeto metiendo todos los valores
 - Get: para obtener el valor de las propiedades
 - Set: para cargar el valor de las propiedades.
 - toString: para mostrar una frase con todos los datos del vehículo

Ejercicio

- Crear ahora un documento en PHP que, por medio de formularios, pida al usuario tipo y peso de dos vehículos diferentes. El formulario debe tener unos campos tipo radio para el tipo de vehículo y un campo de texto para el peso.

- El documento deberá comprobar si los dos vehículos son del mismo tipo o no.

Ejercicio

- Si son del mismo tipo, mostrará por pantalla el vehículo que pese más.
- Si son del mismo tipo y pesan igual, dará un mensaje indicándolo
- Si son de distinto tipo deberá indicar al usuario que no se pueden comparar.

Formulario para pedir datos

Archivo vehiculo.php

Procesamiento.php

```
class vehiculo
{
    private $nom;
    private $tipo;
    private $peso;

    public function __construct ($n, $ti, $pe)
    {
        $this->nom=$n;
        $this->tipo=$ti;
        $this->peso=$pe;
    }

    public function __get($nombre)
    {
        return $this->nombre;
    }

    public function __toString()
    {
        switch ($this->tipo)
        {
            case 'C': $tipo='camión';
                        break;
            case 'M': $tipo='motocicleta';
                        break;
            case 'T': $tipo='turismo';
                        break;
        }

        $cadena = "El vehículo $this->nom es de tipo $tipo y
        pesa $this->peso toneladas";

        return $cadena;
    }
}
```

```
<form action='#' method='post'>
    Nombre de vehículo 1 <input type='text' name='nombre1'>
    <br>
    Tipo de vehículo 1
    <br> Camión <input type='radio' name='tipo1' value='C'>
    Motocicleta <input type='radio' name='tipo1' value='M'>
    Turismo <input type='radio' name='tipo1' value='T'>
    <br>
    Peso del vehículo 1 (en toneladas) <input type='text' name='peso1'>
    <br>
    <hr>
    Nombre de vehículo 2 <input type='text' name='nombre2'>
    <br>
    Tipo de vehículo 2
    <br> Camión <input type='radio' name='tipo2' value='C'>
    Motocicleta <input type='radio' name='tipo2' value='M'>
    Turismo <input type='radio' name='tipo2' value='T'>
    <br>
    Peso del vehículo 2 (en toneladas) <input type='text' name='peso2'>
    <br>
    <input type='submit' name='enviar'>
</form>
```

procesamiento.php segunda parte

```
<?php

require_once('vehiculo.php');

if (isset($_POST['enviar']))
{
    $vehiculo1 = new vehiculo($_POST['nombre1'], $_POST['tipo1'], $_POST['peso1']);
    $vehiculo2 = new vehiculo($_POST['nombre2'], $_POST['tipo2'], $_POST['peso2']);

    if ($vehiculo1->tipo != $vehiculo2->tipo)
    {
        echo "No se pueden comparar los vehículos porque son de diferente tipo";
    }
    else
    {
        if ($vehiculo1->peso > $vehiculo2->peso)
        {
            echo $vehiculo1;
        }
        elseif ($vehiculo2->peso > $vehiculo1->peso)
        {
            echo $vehiculo2;
        }
        else
        {
            echo "Ambos vehículos son del mismo tipo y pesan lo mismo";
        }
    }
}
}
```


Constantes mágicas

- De la misma forma, PHP tiene definidas una serie de **constantes** que nos permitirán encontrar información fácilmente.
- Siempre irán precedidas de `__` y seguidas de `__` (dos guiones bajos)

Constantes mágicas

| Constante | Descripción |
|---------------------------|--|
| <code>__LINE__</code> | Número de línea actual del fichero |
| <code>__FILE__</code> | Ruta completa del fichero |
| <code>__DIR__</code> | Directorio en el que se encuentra el fichero |
| <code>__FUNCTION__</code> | Nombre de la función que se está ejecutando |
| <code>__CLASS__</code> | Nombre de la clase |
| <code>__METHOD__</code> | Nombre del método de la clase |

Herencia de clases

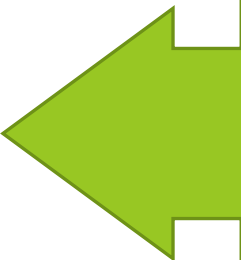
- Para que una clase que creamos pueda heredar las características de otra, utilizaremos la palabra reservada **extends**.

Herencia de clases

```
class niño extends persona
{
    private $nombre_padre;
    private $nombre_madre;
}

$hiijo = new niño();
$hiijo->nombre="Ivan";
$hiijo->apellido= "Pérez";
$hiijo->fecha_nac= "12/12/2001";

echo $hiijo;
```



Clase “niño” que hereda todo de la clase anterior “persona”

En el ejemplo se asignan valores a las propiedades heredadas

- PHP permite que una subclase sobrecargue métodos de la clase superior.
- Si queremos preservar lo que hacía el método en la clase superior utilizaremos la palabra reservada **parent** junto con el operador de resolución de ámbito (**::**)

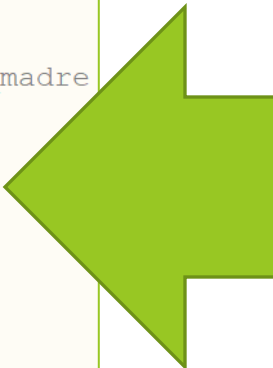
Herencia de clases

```
class niño extends persona
{
    private $nombre_padre;
    private $nombre_madre;

    public function __toString()
    {
        $salida = parent::__toString();
        $salida .= "<br> Su madre se llama $this->nombre_madre
y su padre $this->nombre_padre";
        return $salida;
    }
}

$hiijo = new niño();
$hiijo->nombre="Ivan";
$hiijo->apellido= "Pérez";
$hiijo->fecha_nac= "12/12/2001";

echo $hiijo;
```



En el ejemplo, el método **toString** de la clase **niño** hace lo mismo que hacía el de la clase **persona**, pero además añade el nombre del padre y de la madre

- Partiendo de la clase “animal” que creamos antes, vamos a crear varias clases que hereden de esta:
 - Clase “perro”:
 - La clase “perro” tendrá dos propiedades “raza” y “sexo”.
 - Además, tendrá los siguientes métodos:
 - “ladrar”: mostrará el mensaje “<nombre> dice GUAU”
 - “dormir”: mostrará el mensaje “<nombre> se ha dormido”
 - Clase “delfin”
 - La clase “delfin” tendrá una propiedad “longitud”
 - Además, tendrá los siguientes métodos:
 - “saltar”: mostrará el mensaje “<nombre> está saltando por los aires”
 - “comer”: mostrará el mensaje “<nombre> tiene hambre”
 - Ambas clases deberán tener también sus métodos __set y __get así como __toString

Ejercicio

Clase perro

```
require_once "animal.php";
class perro extends animal
{
    private $raza;
    private $sexo;

    public function __construct ($n='', $c='', $f='', $r='', $s='')
    {
        $this->nombre=$n;
        $this->color=$c;
        $this->f_nac=$f;
        $this->raza=$r;
        $this->sexo=$s;
    }

    public function __set ($pro, $va)
    {
        $this->$pro = $va;
    }

    public function __get ($pro)
    {
        return $this->$pro;
    }

    public function __toString()
    {
        $cadena = "El animal se llama $this->nombre,
        es de color $this->color y es de raza $this->raza";
        return $cadena;
    }

    public function ladrar()
    {
        return "$this->nombre dice GUAU";
    }

    public function dormir ()
    {
        return "$this->nombre se ha dormido";
    }
}
```

Clase delfín

```
require_once "animal.php";
class delfin extends animal
{
    private $longitud;

    public function __construct ($n, $c, $f, $l)
    {
        $this->nombre=$n;
        $this->color=$c;
        $this->f_nac=$f;
        $this->longitud=$l;
    }

    public function __get ($pro)
    {
        return $this->$pro;
    }

    public function __set ($pro, $valor)
    {
        $this->$pro = $valor;
    }

    public function __toString()
    {
        $scad = "El delfín que se llama $this->nombre mide
        $this->longitud metros";
        return $scad;
    }

    public function saltar()
    {
        echo "$this->nombre está saltando por los aires";
    }

    public function comer()
    {
        echo "$this->nombre tiene hambre";
    }
}
```

- Crear un documento PHP que pida al usuario toda la información necesaria para crear un objeto de tipo PERRO y otro de tipo DELFIN. Además, el usuario deberá indicar qué está haciendo el animal.

Ejercicio

- El documento mostrará primero toda la información de cada animal y ejecutará el método que implemente la acción indicada por el usuario.


```

<form action='#' method='get'>
  <b>DATOS DEL PERRO</b>
  <br>
  Nombre: <input type="text" name="nombre_p">
  <br>
  Color: <input type="text" name="color_p">
  <br>
  Fecha de nacimiento: <input type="date" name="fecha_p">
  <br>
  Raza: <input type="text" name="raza">
  <br>
  Sexo: Macho <input type="radio" name="sexo" value='M'>
       Hembra <input type="radio" name="sexo" value='H'>
  ¿Qué está haciendo?
  <br>
  <select name='accion_p'>
    <option value='l' selected> ladrar </option>
    <option value='d'> dormir </option>
  </select>
<br><b>DATOS DEL DELFÍN</b>
  <br>
  Nombre: <input type="text" name="nombre_d">
  <br>
  Color: <input type="text" name="color_d">
  <br>
  Fecha de nacimiento: <input type="date" name="fecha_d">
  <br>
  Longitud: <input type="text" name="long">
  <br>
  ¿Qué está haciendo?<br>
  <select name='accion_d'>
    <option value='s' selected> saltar </option>
    <option value='c'> comer </option>
  </select>
  <input type="submit" name="enviar">
</form>

```

Formulario para
pedir datos

```

if(isset($_GET['enviar']))
{
    if(!$_GET['nombre_p'] || !$_GET['color_p'] || !$_GET['fecha_p']
    || !$_GET['raza'] || !$_GET['sexo'] || !$_GET['accion_p']
    || !$_GET['nombre_d'] || !$_GET['color_d'] || !$_GET['fecha_d']
    || !$_GET['long'] || !$_GET['accion_d'])
    {
        echo "Debes rellenar todos los datos<br>";
        header("refresh:5;URL=formulario_animales.php");
    }
    else
    {
        require_once "perro.php";
        require_once "delfin.php";

        $perrito = new perro($_GET['nombre_p'], $_GET['color_p'],
        $_GET['fecha_p'], $_GET['raza'], $_GET['sexo']);
        $del = new delfin ($_GET['nombre_d'], $_GET['color_d'],
        $_GET['fecha_d'], $_GET['long']);

        if($_GET['accion_p'] == 'l')
        {
            echo $perrito->ladrar();
        }
        else
        {
            echo $perrito->dormir();
        }

        if($_GET['accion_d']=='s')
        {
            $del->saltar();
        }
        else
        {
            $del->comer();
        }
    }
}
}

```

Procesamiento de los datos

PHP Orientado a Objetos