UD-03b: Orientación a Objetos con PHP

Desarrollo Web en Entorno Servidor

Curso 2020/2021

Orientación a objetos en PHP

Utilización de objetos

Ya vimos cómo instanciar un objeto con `new`, y cómo acceder a sus métodos y atributos públicos con el operador `flecha`:

```
$p = new Producto();
$p->nombre = 'Samsung Galaxy S';
$p->muestra();
```

Una vez creado un objeto, puedes utilizar el operador `instanceof` para comprobar si es o no una instancia de una clase determinada.

```
if ($p instanceof Producto)
{
    …
}
```

Orientación a objetos en PHP

Existen una serie de **funciones útiles** para el desarrollo orientado a objetos:

Funciones de utilidad para objetos y clases en PHP5		
Función	**Ejemplo**	**Significado**
get_class	`echo "La clase es: " . get_class($p);`	Devuelve el nombre de la clase del objeto.
class_exists	`if (class_exists('Producto') { $p = new Producto(); … }`	Devuelve true si la clase está definida o false en caso contrario.
get_declared_classes	`print_r(get_declared_classes());`	Devuelve un array con los nombres de las clases definidas.
class_alias	`class_alias('Producto', 'Articulo'); $p = new Articulo();`	Crea un alias para una clase.
get_class_methods	`print_r(get_class_methods('Producto'));`	Devuelve un array con los nombres de los métodos de una clase que son accesibles desde dónde se hace la llamada.
get_object_vars	`print_r(get_object_vars($p));`	Devuelve un array con los nombres de los atributos de una clase que son accesibles desde dónde se hace la llamada.

Orientación a objetos en PHP

Paso de parámetros con tipo

Aún siendo PHP un lenguaje débilmente tipado, sí que **podemos indicar** en las funciones **de qué clase deben ser los objetos que se pasen como parámetros**. Para ello, se especifica el tipo antes del parámetro.

Si cuando se realiza la llamada, el parámetro no es del tipo adecuado, se produce un error. **Sólo funciona con objetos.**

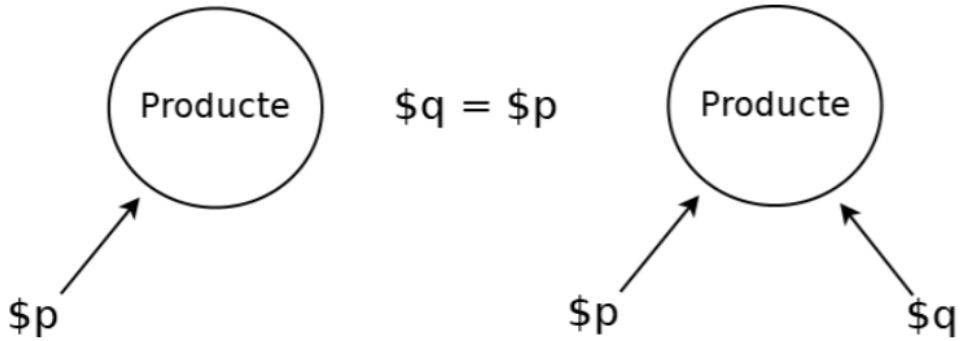
```
public function vendeProducto(Producto $p)
{
    …
}
```

Orientación a objetos en PHP

Referencias a objetos

¿Qué ocurre cuando ejecutamos este código?

```
$p = new Producte();
$q = $p;
$q->desc = 'Samsung Galaxy S';
// ¿Qué valor tiene p->desc?
```



- El código anterior, simplemente crea un nuevo identificador del mismo objeto.
- En cuanto se utilice cualquiera de los identificadores para cambiar el valor de algún atributo, este cambio se vería también reflejado al acceder utilizando el otro identificador.
- Recuerda que aunque haya dos o más identificadores del mismo objeto, todos se refieren a la única copia que se almacena del mismo.

Orientación a objetos en PHP

Copiar objetos

- Por tanto, **no puedes copiar un objeto utilizando el operador = .**
- Si necesitas copiar un objeto, debes utilizar **clone**.
- Al utilizar **clone** sobre un objeto existente, se crea una copia de todos los atributos del mismo en un nuevo objeto.

```
$p = new Producte();
$p->desc= 'Samsung Galaxy S';
$a = clone($p) ';
// ¿Qué valor tiene a->desc?
```

Orientación a objetos en PHP

Personalizar la copia

- Si necesitamos copiar todos los atributos pero con algunas diferencias, **podemos crear un método de nombre __clone** *(otro método mágico)* en la clase.
- Este método **se llamará automáticamente después de copiar todos los atributos** en el nuevo objeto. No sobrescribe la copia sino que añade.
- Cuando un objeto es clonado, se realiza una **copia superficial** de los atributos del objeto, pero **si el objeto contiene otros objetos o referencias, no se hará una copia de estos sino que mantendrá las mismas referencias.** Si deseamos hacer una copia en profundidad deberemos implementarla en le método __clone().

```
class Producto
{
    …
    public function __clone()
    {
        …
    }
}
```

Orientación a objetos en PHP

Ejemplo:

```php
class A
{
    public $Dato;
    public $DatoClaseB;

    public function __construct($dato, $datoClaseB)
    {
        $this->Dato = $dato;
        $this->DatoClaseB = $datoClaseB;
    }

    public function __toString()
    {
        return $this->Dato.' '.$this->DatoClaseB.'<br>';
    }

    public function __clone()
    {
        // Realizamos la copia en profundidad de la agregación.
        $this->DatoClaseB = clone $this->DatoClaseB;
    }
}
```

```php
class B // No implementamos __clone y dejamos copia superficial
{
    public $Dato;

    public function __construct($dato)
    {
        $this->Dato = $dato;
    }

    public function __toString()
    {
        return strval($this->Dato);
    }
}
```

Orientación a objetos en PHP

```php
// ejemplo de uso
$instanciaB = new B(100);
$instanciaA = new A(200, $instanciaB);
$referenciaA = $instanciaA;
$copiaDeA = clone $instanciaA;
$referenciaA->Dato = 414;
$referenciaA->DatoClaseB->Dato = 141;
$copiaDeA->Dato = 313;
$copiaDeA->DatoClaseB->Dato = 131;
echo $instanciaA.$referenciaA.$copiaDeA;
```

Impresión por pantalla:

414 141
414 141
313 131

```php
// otro ejemplo de uso
$instanciaB = new B(100);
$instanciaA = new A(200, $instanciaB);
$referenciaA = $instanciaA;
$referenciaA->Dato = 414;
$referenciaA->DatoClaseB->Dato = 141;
$copiaDeA = clone $instanciaA;
$copiaDeA->Dato = 313;
echo $instanciaA.$referenciaA.$copiaDeA;
```

Impresión por pantalla:

414 141
414 141
313 141

Orientación a objetos en PHP

Comparar objetos

- Para saber la relación exacta entre dos objetos, puedes utilizar == y ===.
- Si utilizas el operador de comparación ==, comparas los valores de los atributos de los objetos.

```
$p = new Producto();
$p->nombre = 'Samsung Galaxy S';
$a = clone($p);

// El resultado de comparar $a == $p da verdadero
```

- Sin embargo, si utilizas el operador ===, el resultado de la comparación será verdadero sólo cuando **las dos variables sean referencias al mismo objeto**.

```
// El resultado de comparar $a === $p da falso
// pues $a y $p no hacen referencia al mismo objeto
```

Ejemplo de getters y setters como métodos mágicos

```php
class Segmento
{
    private $xOrg;
    private $yOrg;
    private $anguloRad;
    private $longitud;

    public function __construct($xOrg, $yOrg)
    {
        $this->xOrg = $xOrg;
        $this->yOrg = $yOrg;
        $anguloRad = 0;
        $longitud = 1;
    }
    public function __get($name)
    {
        switch($name)
        {
            case 'Angulo':
                return $this->anguloRad * 180 / pi();
            case 'Longitud':
                return $this->longitud;
        }
    }
    public function __set($name, $value)
    {
        switch($name)
        {
            case 'Angulo':
                $this->anguloRad = $value * pi() / 180;
                break;
            case 'Longitud':
                $this->longitud = $value;
                break;
        }
    }
    public function __toString()
    {
        return "p=({$this->xOrg},{$this->yOrg})<br>".
               'angRad = '.$this->anguloRad.'<br>'.
               'angGr = '.$this->Angulo.'<br>'. // Se llama a __get(…)
               'long = '.$this->Longitud;
    }
}
```

```php
$s = new Segmento(20,30);
$s->Angulo = 180; // Se llama a __Set('Angulo', 180)
$s->Longitud = 100;

echo $s;
```

p=(20,30)
angRad = 3.1415926535898
angGr = 180
long = 100

EJERCICIOS

- Usando la clase **Contacto** del ejercicio anterior, modifica los **getters** y **setters** e implementarlos como métodos mágicos. Modifica también la clase **Agenda** del ejercicio anterior para añadir **un método de copia** que permita hacer una clonación en profundidad (ten en cuenta que el array que contiene los elementos de la agenda tiene realmente referencias a objetos Contacto). Después crea un script llamado **ud03ejer03.php** que compruebe que todas las modificaciones realizadas funcionan bien.

- Modifica la clase Agenda añadiendo un método llamado ***mostrarLista()***) que muestre la agenda en el navegador y enlaces con el texto "**Ver contacto**" para cada uno de los contactos. Crea un script llamado **ud03ejer04.php** en el que se cree una agenda que contenga inicialmente tres contactos. Cuando el usuario pulse sobre este enlace se le dirigirá a una nueva página llamada **info_contacto.php** que mostrará los datos del contacto seleccionado.