Programación de Servicios y Procesos

Programación Orientada a Objetos en PHP

IES Nervión
Miguel A. Casado Alías

Introducción

- PHP tradicionalmente era un lenguaje de programación estructurada
- A partir de PHP 5 (2004) => Se introduce P.O.O.
- PHP es un lenguaje poco estricto y muy flexible…
  - Permite mezclar programación estructurada y P.O.O.
  - Permite hacer código muy “chapucero”
- Nosotros usaremos PHP para programar de forma orientada a objetos de la manera más pura posible.

Clases

- Cada clase se definirá en un fichero
- Por ejemplo: Person.php

```
<?php
    class Person  {


    }
```

Punto de entrada… ¿No hay método ‘main’?

- No, no hay método main que sirva de punto de entrada
- El punto de entrada es la primera línea del primer fichero que se ejecute. P.ej: index.php

Propiedades y métodos

- Similar a Java, incluyendo la visibilidad
- Si declaramos una propiedad con “var”, se considera que tiene visibilidad pública
- No es necesario indicar lo que devuelven los métodos, aunque a partir de PHP 7 se puede especificar

```php
<?php
    class Person {
        var $name;

        public $height;
        protected $social_insurance;
        private $pinn_number;

        private function get_pinn_number(){
            return $this->pinn_number;
        }
    }
```

$this

- Al igual que en java, tenemos una palabra reservada para referenciar el objeto actual

```
$this->pinn_number;
```

```
$this->update();
```

Getters y Setters

```php
<?php
    class Person {
        private $name;

        function set_name($new_name) {
            $this->name = $new_name;
        }

        function get_name() {
            return $this->name;
        }
    }
?>
```

Cómo hacer uso de nuestras clases

- En el fichero en el que queramos usar una clase (por ejemplo, en index.php) deberemos usar una de las siguientes sentencias:

```
<?php include "Person.php"; ?>

<?php include_once "Person.php"; ?>

<?php require "Person.php"; ?>

<?php require_once "Person.php"; ?>
```

- _once indica que solo se incluirá una vez el fichero
- Las sentencias require dan error (detienen ejecución) si no se puede incluir el fichero. Las include solo dan warning

Instanciar objetos e invocar sus métodos

```php
<?php

require_once "Person.php";

$human = new Person();

$human->set_name("Michael");
echo $human->get_name();
```

Constructores

- Son opcionales
- No se pueden sobrecargar debido a que PHP es poco estricto y soporta ejecutar una función (o método) enviándole un número de parámetros diferente al número de parámetros con el que se declaró

```php
<?php
    class Person {
        private $name;

        function __construct($persons_name) {
            $this->name = $persons_name;
        }
```

- Y para instanciar:

```php
<?php
$human = new Person("Michael");
```

Herencia

- Se usa la palabra reservada extends
- Se pueden sobreescribir los métodos de la clase padre

```
class Employee extends Person
{
    function __construct($employee_name) {
        $this->set_name($employee_name);
    }
}
```

parent

- Con la palabra reservada parent podemos acceder a métodos de la clase padre
- Es parecido a super en java

```
parent::set_name($new_name);
```

- Usamos :: en lugar de ->
  - :: se usa para acceder a elementos estáticos
  - -> se usa para acceder a elementos que son instancias
  - Más info:

    http://stackoverflow.com/questions/3173501/whats-the-difference-between-double-colon-and-arrow-in-php

self

- Se usa para hacer referencia a la CLASE actual
- Recuérdese que $this hace referencia al OBJETO actual

```
<?php
class A {
    public static function saludo() {
        echo “HOLA";
    }
    public static function test() {
        self::saludo();
    }
}
```

Interfaces

- Se usa la palabra reservada interface
- Todos los métodos deben ser públicos
- Para implementar la interfaz se usará la palabra implements

```
interface ITemplate
{
    public function setVariable($name, $var);
}


class Template implements ITemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }
}
```

Métodos mágicos

- Son métodos que podemos definir en nuestras clases, los cuales se invocarán cuando se produzca algún evento determinado
- Los métodos mágicos siempre empiezan por __ , así que no deberíamos comenzar los nombres de métodos no mágicos por __
- Los métodos mágicos existentes son:
  - __construct()
  - __destruct()
  - __call()
  - __callStatic()
  - __get()
  - __set()
  - __isset()
  - __unset()
  - __sleep()
  - __wakeup()
  - __toString()
  - __invoke()
  - __set_state()
  - __clone()
  - __debugInfo()

Métodos mágicos (II)

- __construct() y __destruct() son llamados cuando se crea un objeto y cuando se destruye respectivamente
- __call() y __callStatic() serán llamados cuando se intenta hacer una llamada a un método no accesible. Si el método no accesible es estático, se llamará a __callStatic(), en caso contrario a __call()
- __get() y __set() se llaman cuando se intenta leer (get) o escribir (set) propiedades inaccesibles
- __isset() y __unset() son lanzados cuando se intentan llamadas a isset(), empty() o unset() sobre propiedades inaccesibles
- __sleep() se llama antes de hacer una serialización del objeto, y __wakeup() al hacer una deserialización del mismo

Métodos mágicos (III)

- __toString() debe devolver un string, y será llamado cuando el objeto se trate como un string (por ejemplo al hacer echo de él)
- __invoke() es llamado cuando se llama al objeto como si fuera una función (p.ej: `$miObjeto(7);`)
- __set_state() es llamado al invocar a var_export() con nuestro objeto como parámetro
- __clone() se llama cuando se hace una clonación del objeto (p.ej: `$objB = clone $objA;`)
- __debugInfo() se invoca cuando se hace una llamada a var_dump() pasándole nuestro objeto

Métodos mágicos (IV)

- Más información sobre métodos mágicos:
  - http://php.net/manual/es/language.oop5.magic.php
  - http://lornajane.net/posts/2012/9-magic-methods-in-php
  - https://diego.com.es/metodos-magicos-en-php

Ejercicio

- Hacer una clase que contenga métodos para realizar distintas operaciones con un texto, como por ejemplo:
  - Indicar el número de veces que aparece una palabra
  - Indicar las posiciones en que aparece una palabra
  - Sustituir una palabra por otra
  - Sustituir la palabra de la posición x por la de la posición y
  - Y algunas más que se te ocurran…
- Haciendo uso de la clase anterior, realizar una aplicación web que permita al usuario introducir un texto y realizar alguna operación con él, mostrándole el resultado.