PROGRAMACIÓN ORIENTADA A OBJETOS EN PHP

T 5.3

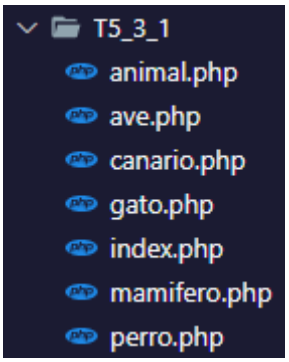
David Rodríguez Jácome

2º DAW Desenvolvemento Web en Contorno Servidor

En esta actividad haremos uso de la programación orientada a objetos para crear dos programar y ver su funcionamiento. Seguiremos dos ejemplos del libro de Luis Sánchez.

1. Programa de animales.

En este apartado crearemos un programa para emular el comportamiento de varios animales. En primer lugar, creamos los archivos que necesitamos:



Seguidamente creamos la clase padre Animal, que será la clase de la que hereden otras. Añadiremos un atributo privado “sexo”, un constructor para dar valor al atributo de la clase, un método toString para pasar la clase a una cadena de texto, un método getter para tomar el valor que almacena el atributo de la clase, y dos métodos “comer” y “dormir” para poder emular comportamientos.

animal.php

Tareas > T5 > T5_3 > T5_3_1 > animal.php > PHP Intelephense > Animal

```php
<?php
// Clase padre "Animal".
class Animal    {
    // Atributos.
    private $sexo;

    // Constructor de la clase.
    public function __construct($sexo)  {
        if (!isset($sexo)) {
        $this->sexo = "macho";
        } else {
        $this->sexo = $sexo;
        }
    }

    // Este método convierte la clase en una cadena.
    public function __toString()    {
        return "Sexo: $this->sexo";
    }

    // Método para tomar el valor del atributo de la clase.
    public function getSexo() {
        return $this->sexo;
    }

    // Método para representar una acción de la clase.
    public function comer($comida) {
        return "Estoy comiendo $comida" . ".";
    }

    // Método para representar una acción de la clase.
    public function dormir() {
        return "Estoy durmiendo.";
    }
}
```

La subclase Mamífero hereda de la clase Animal, lo que significa que puede usar los atributos (a través del método get) y métodos de la clase de la que hereda, además de poder usar los suyos propios.

mamifero.php

Tareas > T5 > T5_3 > T5_3_1 > mamifero.php > PHP Intelephense > Mamifero

```php
<?php
// Subclase hija que hereda de la clase padre "Animal".
include_once "animal.php";

class Mamifero extends Animal {
    // Constructor que extiende del constructor de la clase padre "Animal".
    public function __construct($sexo)  {
        parent::__construct($sexo);
    }

    // Constructor de la propia clase.
    public function amamantar() {
        if ($this->getSexo()=="macho") {
            return "Soy macho, no puedo amamantar.";
        } else {
            return "soy hembra, puedo amamantar.";
        }
    }

    // Constructor de la propia clase.
    public function cuidarCrias() {
        return "Estoy cuidando crías.";
    }

    // Constructor de la propia clase.
    public function caminar() {
        return "Estoy caminando.";
    }
}
```

Lo mismo ocurre con las subclases Ave, Gato, Perro y Canario:

ave.php

Tareas > T5 > T5_3 > T5_3_1 > ave.php > PHP Intelephense > Ave > asearse

```php
<?php
include_once "animal.php";

// Subclase hija que hereda de la clase padre "Animal".
class Ave extends Animal {
    public function __construct($sexo) {
        parent::__construct($sexo);
    }

    // Constructor de la propia clase.
    public function asearse() {
        return "Me estoy limpiando las plumas.";
    }

    // Constructor de la propia clase.
    public function volar() {
        return "Estoy volando.";
    }

    // Constructor de la propia clase.
    public function ponerHuevo() {
        if(!$this->getSexo() == "macho") {
            return "Soy un macho, no puedo poner huevos.";
        } else {
            return "He puesto un huevo.";
        }
    }
}
```

gato.php

Tareas > T5 > T5_3 > T5_3_1 > gato.php > ...

```php
<?php

include_once "animal.php";

class Gato extends Animal {
    private $raza;

    public function __construct($sexo, $raza) {
        parent::__construct($sexo);
        if (isset($raza)) {
            $this->raza = $raza;
        } else {
            $this->raza = "Siamés";
        }
    }

    public function __toString() {
        return parent::__toString() . "<br>Raza: $this->raza";
    }

    public function maullar() {
        return "Miau.";
    }

    public function ronronear() {
        return "Rondoneo.";
    }

    public function comer($comida) {
        if ($comida == "pescado") {
            return "Estoy comiendo pescado.";
        } else {
            return "No me gusta otra cosa que el pescado.";
        }
    }

    public function pelear($rival) {
        if ($this->getSexo() == "Hembra") {
            return "Soy hembra; no peleo.";
        } else if ($rival->getSexo() == "Hembra") {
            return "No peleo contra hembras.";
        } else {
            return "Voy a pelear";
        }
    }

    public function asearse() {
        return "Me estoy aseando.";
    }

    public function cazar() {
        return "Estoy cazando ratones.";
    }
}
```

perro.php

Tareas > T5 > T5_3 > T5_3_1 > perro.php > ...

```php
<?php

include_once "mamifero.php";

class Perro extends Mamifero {
    private $raza;

    public function __construct($sexo, $raza) {
        parent::__construct($sexo);
        if (isset($raza)) {
            $this->raza=$raza;
        }else{
            $this->raza="Pastor alemán.";
        }
    }

    public function __toString()    {
        return parent::__toString()."<br>Raza: $this->raza";
    }

    public function ladrar() {
        return "Guau guau.";
    }

    public function rascarse() {
        return "Me estoy rascando.";
    }

    public function cazar() {
        return "Estoy cazando.";
    }
}
```

canario.php

Tareas > T5 > T5_3 > T5_3_1 > canario.php > ...

```php
<?php

include_once "ave.php";

class Canario extends Ave {
    private $color;

    public function __construct($sexo, $color)  {
        parent::__construct($sexo);
        if (!isset($color)) {
            $this->color = $color;
        } else {
            $this->color = "Amarillo";
        }
    }

    public function __toString()    {
        return parent::__toString()."<br>Color: $this->color";
    }

    public function cantar() {
        return "Pío pío pío";
    }

    public function bucear() {
        return "Los canarios no bucean";
    }
}
```

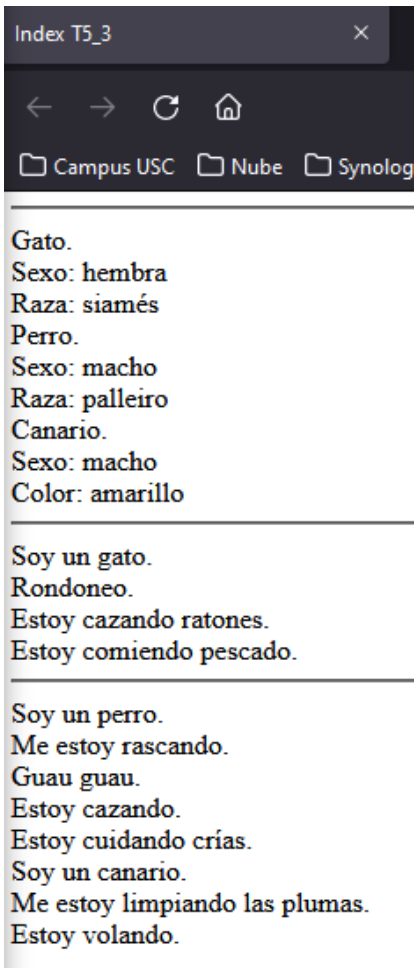
Una vez tengamos todas las clases, codificaremos un archivo Index donde llamaremos a cada clase para instanciar (crear) los objetos de las mismas y definir su comportamiento mediante sus métodos:

index.php

Tareas > T5 > T5_3 > T5_3_1 > index.php > ...

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Index T5_3</title>
</head>
<body>
    <?php

    include_once "gato.php";
    include_once "perro.php";
    include_once "canario.php";

    $gato = new Gato("hembra","siamés");
    $perro = new Perro("macho", "palleiro");
    $canario = new Canario ("macho","amarillo");

    echo "<hr>Gato: " . $gato . "<br>";
    echo "Perro: " . $perro . "<br>";
    echo "Canario: " . $canario . "<br><hr>";

    echo "Soy un gato.<br>";
    echo $gato -> ronronear() . "<br>";
    echo $gato -> cazar() . "<br>";
    echo $gato -> comer("pescado") . "<br>";

    echo "<hr>Soy un perro.<br>";
    echo $perro -> rascarse() . "<br>";
    echo $perro -> ladrar() . "<br>";
    echo $perro -> cazar() . "<br>";
    echo $perro -> cuidarCrias() . "<br>";

    echo "Soy un canario.<br>";
    echo $canario -> asearse() . "<br>";
    echo $canario -> volar() . "<br>";
    ?>
</body>
</html>
```

El resultado de la ejecución de este programa será el siguiente:



2. Programa de dados.

En este ejercicio haremos una aplicación que use una clase Dado para usar y tirar cinco dados de 6 caras de un cubilete. La estructura de los archivos será la siguiente:

La clase Dado será la que tenga las características y comportamiento disponible de los dados:

dado.php

Tareas > T5 > T5_3 > T5_3_2 > dado.php > PHP Intelephense > Dado

```php
<?php

// Clase con el objeto Dado.
class Dado {
    // Variable private static para definir que los
    // objetos que se creen de la clase Dado tengan 6 caras.
    private static $caras = array("1","2","3","4","5","6");
    // Variable para contar cuántas tiradas se han hecho
    // en una misma ejecución del programa.
    private static $tiradasTotales = 0;
    private $cara;

    // Método getter que devuelve cuántas tiradas se han hecho.
    public static function getTiradasTotales() {
        return Dado::$tiradasTotales;
    }

    public static function setTiradasTotales($tiradasTotales) {
        self::$tiradasTotales = $tiradasTotales;
    }

    // Método para tirar un dado.
    public function tira() {
        $this->cara = self::$caras[rand(0,5)];
        self::$tiradasTotales++;
    }

    // Método para indicar qué cara del dado salió.
    public function caraDado() {
        return $this->cara;
    }
}
```

Seguidamente, codificaremos el programa principal en el archivo “Index” que será el encargado de la ejecución. La primera imagen es la parte PHP y la segunda la parte HTML visible por pantalla:

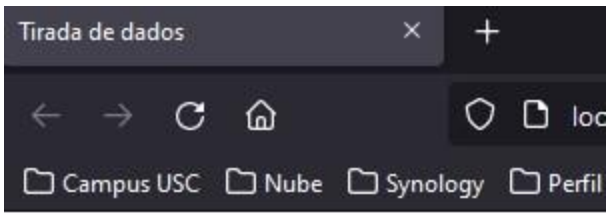
index.php

Tareas > T5 > T5_3 > T5_3_2 > index.php > html > body

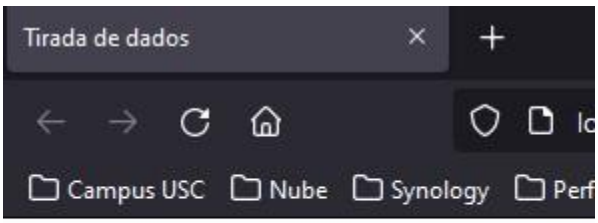
```php
<?php
// Iniciamos una sesión.
session_start();

// Incluímos la clase Dado.
include_once "dado.php";

// Creamos 5 dados serializados.
if (!isset($_SESSION["misDados"])) {
    $_SESSION["misDados"]=serialize(array(new Dado(),new Dado(),new Dado(),new Dado(),new Dado()));
}

if(!isset($_SESSION["tiradasTotales"])) {
    $_SESSION["tiradasTotales"] = 0;
}

$misDados = unserialize($_SESSION["misDados"]);
Dado::setTiradasTotales($_SESSION["tiradasTotales"]);
?>
```

```php

<!-- HTML con la visualización del resultado por pantalla.-->
<!DOCTYPE html>
<html>

<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Dados de Poker</title>
</head>

<body>
    <?php
    // Llamamos a los métodos para lanzar y detallar
    // el resultado de la tirada de cada dado.
    echo "Resultado de esta tirada de dados: ";
    foreach ($misDados as $dado) {
        $dado->tira();
        echo ($dado->caraDado()) . " ";
    }

    echo "<br>Tiradas de dados totales: " . (Dado::getTiradasTotales());
    echo "<br>Tiradas de cubilete: " . (Dado::getTiradasTotales()/5);

    $_SESSION["misDados"] = serialize($misDados);
    $_SESSION["tiradasTotales"] = Dado::getTiradasTotales();
    ?>
</body>
</html>
```

Finalmente, el resultado por pantalla es el que sigue:



Resultado de esta tirada de dados: 1 5 2 5 6
Tiradas de dados totales: 10
Tiradas de cubilete: 2



Resultado de esta tirada de dados: 4 5 4 3 5
Tiradas de dados totales: 35
Tiradas de cubilete: 7

BIBLIOGRAFÍA/WEBGRAFÍA

- Sánchez González, Luis J. (2016): *Aprende PHP con ejercicios*, ed. Leanpub, Málaga.