Guía Práctica de Ejercicios Nº 5

En nuestra quinta práctica daremos comienzo a la programación orientada a objetos, definiendo algunos conceptos fundamentales del paradigma POO, mencionando algunas de las características del mismo y luego pasaremos a la implementación de la POO en PHP versión 5.

Objetivo: Comprender los conceptos básicos de la programación orientada a objetos, definiendo las clases y objetos con sus atributos y métodos. Definir, desarrollar y utilizar los métodos constructores, destructores y toString. Entender y usar parámetros por defecto en los métodos de una clase. Utilizar objetos como parámetros en los métodos de una clase.

Introducción: En nuestra actualidad vemos que el desarrollo de Internet ha sido inminente y con ello las aplicaciones Web, por lo tanto, se hace indispensable el uso de un lenguaje que permita desarrollar aplicaciones Web como PHP, entre otros. Teniendo en cuenta la situación anterior es que veremos la como desarrollar aplicaciones Web que se ejecutarán del lado del servidor utilizando uno de los mejores lenguajes del ambiente del Software Libre. Teniendo en cuenta la magnitud del proyecto a desarrollar veremos la necesidad de emplear el paradigma orientado a objetos con PHP; que nos dará la posibilidad de diseñar, desarrollar y mantener el Software de manera profesional utilizando un paradigma antes mencionado.

Construcciones (palabras clave), variables globales y funciones que estudiaremos en esta unidad.

Construcciones del lenguaje con las que trabajaremos.

class nos permite crear una clase en PHP que es una colección de variables (atributos) y funciones (métodos) que trabajan con estas variables.

public nos permite definir atributos y métodos para que sean accedidos desde cualquier lugar.

private nos permite definir atributos y métodos para que sean accedidos desde la clase donde fueron definidos.

protected nos permite definir atributos y métodos para que sean accedidos desde la clase donde fueron definidos y sus descendientes.

final nos permite definir atributos y métodos para que sean accedidos desde cualquier lugar pero no pueden ser reescritos por sus descendientes.

new nos permite instanciar un objeto de acuerdo a una clase.

Ejercicios

Utilizar los conceptos de programación Web y de base de datos

1) – Escriba un script PHP para definir una clase de nombre HolaMundo que implemente un método de nombre saludar() para devuelva un string (Hola Mundo!) de saludo.

HolaMundo.php

```
<?php
// Se define la clase HolaMundo (HolaMundo.php)
class HolaMundo
{
	// Se define el atributo privado $_saludo que contiene el mensaje de saludo
	private $_saludo = 'Hola Mundo!';
	// Se implementa el método saludar para retornar el mensaje de saludo
	public function saludar()
	{
		return $this->_saludo;
	}
}
// Se crea (instancia) el objeto $oHolaMundo de la clase HolaMundo
$oHolaMundo = new HolaMundo();
// Se llama al método saludar para obtener el mensaje de saludo y mostrarlo en pantalla
echo $oHolaMundo->saludar();
?>
```

2) - Realice un script de nombre Persona.php que defina y use una clase (Persona) con los siguientes atributos y métodos:

Atributos

(privado) tipoDocumento
(privado) numeroDocumento
(privado) apellidos
(privado) nombres
(privado) domicilio

(privado) telefono
(privado) telefonoTrabajo
(privado) telefonoMovil
(privado) correoElectronico

Métodos

(público) __construct()
(público) cambiarDomicilio()
(público) cambiarNumeroTelefono()
(público) cambiarNumeroTelefonoMovil()
(público) cambiarNumeroTelefonoTrabajo()
(público) cambiarCorreoElectronico()

Dentro del mismo script cree un objeto que use todos los métodos que se definieron.

Persona.php

```
<?php
class Persona
{
	// Se definen los atributos de la clase
	private $_tipoDocumento;
	private $_numeroDocumento;
	private $_apellidos;
	private $_nombres;
	private $_domicilio;
	private $_telefono;
	private $_telefonoTrabajo;
	private $_telefonoMovil;
	private $_correoElectronico;
	// Se implementan los métodos de la clase
	public function cambiarDomicilio($domicilio)
	{
		$this->_domicilio = $domiclio;
	}
	public function cambiarNumeroTelefono($telefono)
	{
```

```
            $this->_telefono = $telefono;
        }
        public function cambiarNumeroTelefonoMovil($telefonoMovil)
        {
            $this->_telefonoMovil = $telefonoMovil;
        }
        public function cambiarNumeroTelefonoTrabajo($telefonoTrabajo)
        {
            $this->_telefonoTrabajo = $telefonoTrabajo;
        }
        public function cambiarCorreoElectronico($correoElectronico)
        {
            $this->_correoElectronico = $correoElectronico;
        }
}
$oPersona = new Persona();
$oPersona->cambiarDomicilio('San Martín 1222');
$oPersona->cambiarNumeroTelefono('4221122');
$oPersona->cambiarNumeroTelefonoMovil('154232322');
$oPersona->cambiarNumeroTelefonoTrabajo('4212221');
$oPersona->cambiarCorreoElectronico('correoelectronico@gmail.com');
?>
```

3) – Confeccionar una clase CabeceraPagina (CabeceraPagina.php) que permita mostrar un título, indicarle si queremos que aparezca centrada, a derecha o izquierda. Utilizar un constructor para inicializar los dos atributos.

CabeceraPagina.php

```
<?php
class CabeceraPagina
{
        private $_titulo;
```

```
	private $_alineacionTitulo;
	private $_cabeceraPagina = '';
	public function __construct($titulo, $alineacionTitulo)
	{
		$this->_titulo = $titulo;
		$this->_alineacionTitulo = $alineacionTitulo;
	}
	private function _armarCabeceraPagina()
	{
		$this->_cabeceraPagina = “<h1 align='”.$this->_alineacionTitulo.”'>”.$this->_titulo.”</h1>”;
	}
	public function mostarCabeceraPagina()
	{
		$this->_armarCabeceraPagina();
		return $this->_cabeceraPagina;
	}
}
?>
```

4) – Confeccionar una clase PiePagina (PiePagina.php) que permite mostrar el/los años, el autor y datos de contacto (dirección Web y correo electrónico). Indicarle si queremos que aparezca centrado, a derecha o izquierda. Utilizar un constructor para inicializar los dos atributos.

PiePagina.php

```
<?php
class PiePagina
{
	private $_autor;
	private $_webAutor;
	private $_emailAutor;
	private $_anioDesde;
	private $_anioHasta;
```

```
	private $_alineacion;
	private $_piePagina;
	public function __construct($autor, $webAutor, $emailAutor, $anioDesde, $anioHasta, $alineacion)
	{
		$this->_autor = $autor;
		$this->_webAutor = $webAutor;
		$this->_emailAutor = $emailAutor;
		$this->_anioDesde = $anioDesde;
		$this->_anioHasta = $anioHasta;
		$this->_alineacion = $alineacion;
	}
	private function _armarPiePagina()
	{
		$this->_piePagina = "<p align='".$this->_alineacion."'>&copy;".$this->_anioDesde." - ".$this->_anioHasta." ".$this->_autor."<br/>".$this->_webAutor." / ".$this->_emailAutor."</p>";
	}
	public function mostrarPiePagina()
	{
		$this->_armarPiePagina();
		return $this->_piePagina;
	}
}
?>
```

5) – Crear otro script PHP (index.php) que use las clases definidas en los puntos 3 y 4 de la presente guía práctica.

index.php

```
<?php
require_once 'CebeceraPagina.php';
require_once 'PiePagina.php';
```

```
$oCabeceraPagina = new CabeceraPagina('Mi página personal', 'center');

echo $oCabeceraPagina->mostarCabeceraPagina();

$oPiePagina = new PiePagina('Apellidos, Nombres', 'http://www.apellidosnombres.com.ar',
'apellidosnombres@gmail.com', '2009', '2010', 'center');

echo $oPiePagina->mostrarPiePagina();

?>
```

6) – Editar la clase Persona del punto 2 para agregar el atributo privado $_hermanos de tipo array() y un método que permita agregarlos. Pruebe dentro del mismo scirpt agregar hermanos a un objeto persona.

Persona.php

```
<?php

class Persona

{

        // Se definen los atributos de la clase

        private $_tipoDocumento;

        private $_numeroDocumento;

        private $_apellidos;

        private $_nombres;

        private $_domicilio;

        private $_telefono;

        private $_telefonoTrabajo;

        private $_telefonoMovil;

        private $_correoElectronico;

        private $_hermanos = array();

        // Se implementan los métodos de la clase

        public function cambiarDomicilio($domicilio)

        {

                $this->_domicilio = $domiclio;

        }

        public function cambiarNumeroTelefono($telefono)

        {
```

```
            $this->_telefono = $telefono;
        }
        public function cambiarNumeroTelefonoMovil($telefonoMovil)
        {
            $this->_telefonoMovil = $telefonoMovil;
        }
        public function cambiarNumeroTelefonoTrabajo($telefonoTrabajo)
        {
            $this->_telefonoTrabajo = $telefonoTrabajo;
        }
        public function cambiarCorreoElectronico($correoElectronico)
        {
            $this->_correoElectronico = $correoElectronico;
        }
        public function agregarHermano(Persona $oHermano)
        {
            $this->_hermanos[] = $oHermano;
        }
}
$oPersona = new Persona();
$oHermanoPersona = new Persona();
$oHermanaPersona = new Persona();
$oPersona->agregarHermano($oHermanoPersona);
$oPersona->agregarHermano($oHermanaPersona);
?>
```

7) – Para los puntos 2, 3 y 4 agregar el método __toString().

Persona.php

```
<?php
class Persona
{
```

```
        …
        public function __toString()
        {
                return $this->_apellidos.", ".$this->_nombres;
        }
}
?>
```

CabeceraPagina.php

```
<?php
class CabeceraPagina
{
        …
        public function __toString()
        {
                return $this->_titulo;
        }
}
?>
```

PiePagina.php

```
<?php
class PiePagina
{
        …
        public function __toString()
        {
                return $this->_autor." / ".$this->_webAutor." / ".$this->_emailAutor;
        }
}
?>
```

Anotaciones de los Ejercicios


Gugler