

Programación III

PHP

Clase 2

Temas a Tratar

Programación del lado del Servidor

Funciones propias

Incluir/Requerir archivos

Clases y objetos

Funciones propias en PHP

(1/2)

La declaración de una función comienza con la palabra ***function***.

El nombre de la función puede empezar con una letra o guión bajo (_), no con números.

Los nombres de las funciones NO son case-sensitive.

```
function NombreFuncion() {  
    //código  
}
```

Funciones propias en PHP

(2/2)

Las funciones pueden recibir parámetros.

```
function NombreFuncion($par_1, $par_2, ..., $par_n) {  
    //código  
}
```

Las funciones pueden retornar valores.

```
function NombreFuncion() {  
    return $valor;  
}
```

Los parámetros pueden tener valores por default.

```
function NombreFuncion($par_1, $par_2 = "valor") {  
    //código  
}
```

Temas a Tratar

Programación del lado del Servidor

Funciones propias

Incluir/Requerir archivos

Clases y objetos

Incluir archivos en PHP

La declaración *include* (o *require*) copia todo el código existente del archivo especificado dentro del archivo que posee dicha declaración.

La declaración *include* y *require* son idénticas, excepto en caso de falla.

require producirá un error fatal (E_COMPILE_ERROR) y frenará el script.

include sólo producirá una advertencia (E_WARNING) y el script continuará.

```
<?php  
include "nombre_archivo";  
  
require "nombre_archivo";  
?>
```

Temas a Tratar

Programación del lado del Servidor

- Funciones propias

- Incluir/Requerir archivos

- Clases y objetos

Clases y objetos

La sintaxis básica para declarar una clase en PHP

```
class NombreClase
{
    //Atributos.
    //Métodos.
}
```

La sintaxis básica para declarar miembros de una clase (atributos - métodos)

```
//Atributos.
[Modificadores] $nombreAtributo;
//Métodos.
[Modificadores] function NombreMetodo([parametros])
{ ... }
```


Clases

```
class NombreClase
{
    //Atributos.
    private $_attr1;
    protected $_attr2;
    var $attr3;
    public static $attr4;

    //Constructor
    public function __construct() { // código }

    //Métodos.
    private function Func1($param) { //código }
    protected function Func2() { //código }
    public function Func3() { //código }
    public static function Func4() { //código }
}
```

Objetos

La sintaxis básica para declarar un objeto en PHP

```
//Creo el objeto.  
$nombreObj = new NombreClase();
```

El operador -> es utilizado para acceder a los miembros de instancia de la clase.

```
//Métodos de instancia.      //Atributos de instancia.  
$nombreObj->Func3();          $nombreObj->attr3;
```

El operador :: es utilizado para acceder a los miembros estáticos de la clase.

```
//Métodos de clase.          //Atributos estáticos  
NombreClase::Func4();        NombreClase::$attr4;
```

\$this

La sintaxis para acceder a un atributo o método de una instancia en PHP

```
class ClaseBase {  
    ...  
    public function __construct($id, $nombre) {  
        //Inicializar variables aquí  
        if($this->validar($id)) {  
            $this->id=$id;  
            $this->nombre=$nombre;  
        }  
    }  
    public function validar($id) {  
        //realiza una validación  
    }  
}
```

Herencia

En PHP se indica herencia a partir de ***extends***.

```
class ClaseBase {  
    public function __construct() {  
        //Inicializar variables aquí  
    }  
}
```

```
class ClaseDerivada extends ClaseBase {  
    public function __construct() {  
        parent::__construct();  
        //Inicializar variables propias aquí  
    }  
}
```

Polimorfismo

En PHP cualquier método puede ser modificado en sus clases derivadas.

```
class ClaseBase {  
    public function Saludar() {  
        return "Hola";  
    }  
}
```

```
class ClaseDerivada extends ClaseBase {  
    public function Saludar() {  
        return parent::Saludar() . " ". "mundo";  
    }  
}
```

Interfaces

Las interfaces en PHP sólo pueden contener declaraciones de métodos.

```
interface IInterfaz{  
    function Metodo();  
}
```

Y se implementan con *implements*.

```
class Clase implements IInterfaz {  
    public function Metodo() {  
        //Implementación aquí  
    }  
}
```

Clases abstractas

Las clases abstractas pueden contener atributos y métodos, pero sólo ellas pueden contener métodos con el modificador **abstract**

```
abstract class ClaseAbstracta {  
    public abstract function Metodo();  
}
```

```
class ClaseDerivada extends ClaseAbstracta {  
    public function Metodo() {  
        //Implementación aquí  
    }  
}
```



Ejercitación

Ejercicios de Programación

Realizar los ejercicios de la guía.

Aplicar las recomendaciones estándares PSR-1.

<https://www.php-fig.org/psr/psr-1/>