

Programación III

PHP

Clase 2

Maximiliano Neiner

Temas a Tratar

- Programación del lado del Servidor

Temas a Tratar

- Programación del lado del Servidor
 - Funciones propias
 - Incluir/Requerir archivos
 - Clases y objetos

Funciones propias en PHP (1/3)

- 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.
- Se las puede definir con tipado débil o tipado fuerte.

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

Funciones propias en PHP (2/3)

- 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  
}
```

Funciones propias en PHP (3/3)

- Tipado fuerte:
- Se les indica tipos de parámetros y tipo de retorno.

```
function nombreFuncion(tipo $param) : tipo_retorno{  
    return tipo_retorno  
}
```

- Unión de tipos

```
function nombreFuncion(tipo1 | tipo2 $param)  
    : tipo_retorno1 | tipo_retorno2{  
  
    return tipo_retorno1;  
}
```

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([parámetros])
{ ... }
```

Classes

```
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;
```

Herencia

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

```
class ClaseBase {  
    public function __construct(){  
        //Inicializar variables aquí  
        //$this representa al objeto actual  
    }  
}
```

```
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():string{  
        return "Hola";  
    }  
}
```

```
class ClaseDerivada extends ClaseBase {  
    public function saludar():string{  
        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  
    }  
}
```


Namespaces

- Deben ser la primera línea de código.

```
namespace Identificador;
```

- Pueden contener clases, interfaces, funciones o constantes. Se utilizan:
- Con nombre completamente cualificado:

```
Identificador\elemento
```

- Con nombre abreviado:

```
use identificador\{clase1, clase2, interface,  
...etc}  
//elemento (clase1 o clase2 o interface o ...)
```



Ejercitación

Ejercicios de Programación

- Realizar los ejercicios de la guía.
- Aplicar las recomendaciones estándares PSR-1.

