Desarrollo web en entorno servidor

UT 1.1 – Arquitectura y herramientas web

4 – PHP – Funciones – Variables globales – Objetos

IES Clara del Rey – Madrid

PHP – Funciones

Una función es un bloque de código reutilizable que puede llamarse para llevar a cabo una tarea específica.

Las funciones pueden existir fuera de las clases, y se usan para hacer el código más modular y reutilizable.

Cuando las funciones se definen dentro de clases, sí se llaman métodos.

Sintaxis básica de una función:

```
<?php
    function nombreDeLaFuncion() {
        // Bloque de código
        return $resultado; #
    }
?>
```

PHP – Funciones – Parámetros

Las funciones pueden declarar parámetros, que se escriben con $ como las variables:

function nombreDeLaFuncion($parametro1, $parametro2)

Desde PHP 7, se puede definir tipo en parámetros. Si no se cumple el requisito al llamar a una función, se lanza un error "TypeError".

function nombreDeLaFuncion(<tipo> $param1, <tipo> $param2)

Para que PHP lance errores de tipo, hay que activar los tipos estrictos:

declare(strict_types=1);

PHP – Funciones – Parámetros

Los tipos admitidos son: int, float, string, bool, array, callable (como una función anónima, por ejemplo), object. Ejemplo:

function fibonacci(int $numero) { ... }

Respecto a objetos, admite "object", o cualquier clase o interfaz. Por ejemplo, si se tiene una interfaz "Operable", se puede definir:

function anestesiar(Operable $paciente) { ... }

Paciente puede ser cualquier clase que implemente la interfaz Operable.

Si no se han activado los tipos estrictos, PHP puede informar de los potenciales errores con warnings, pero siempre intentará realizar la conversión de tipos.

PHP – Funciones – Tipos de unión

En PHP 8 se introdujeron los tipos de unión, que permiten indicar que un parámetro puede ser de más de un tipo. Ejemplo:

```
function nombreDeLaFuncion(int|string $parametro)
```

Esto indica que la función admite tanto enteros como strings para la entrada, pero no otros tipos, como float o bool.

Para evaluar el tipo de dato recibido pueden usarse las funciones "is_", como is_int, is_float, is_string, is_array o is_object.

"is_numeric", devuelve true si el parámetro que recibe es un número, o si es string con valor numérico. Estas dos llamadas devuelven true:

```
is_numeric(20458.25)
is_numeric("2568")
```

PHP – Funciones – Nullables

PHP también permite que se definan parámetros que admiten cierto tipo, pero que también pueden recibir null. Se utiliza una interrogación delante del tipo, para indicar que es nullable, y la función is_null para comprobar si un parámetro es null :

```
function showAge(?int $age){
        if (is_null($age)){
                echo "Edad no proporcionada";
        } else {
                echo "Edad: {$edad}";
        }
}
```

Cuidado, en PHP la interrogación va antes del tipo, pero en otros lenguajes se escribe detrás, como, por ejemplo, C#

PHP – Funciones – Valores por defecto

Se puede declarar un valor por defecto para un parámetro. Si se llama a una función sin el parámetro, este tomará el valor por defecto indicado.

Ejemplo:

```
function saludar(string $nombre = "José Luis"){
        echo "Hola, {$nombre}.";
}

saludar();          // Muestra "Hola, José Luis."
saludar("Ana");     // Muestra "Hola, Ana."
```

PHP – Funciones – Devolución de valores

Una función no tiene por qué devolver un valor. Por convención, aunque no devuelva valor, se seguirá llamando función, y no procedimiento, como en otros lenguajes, como Pascal o SQL, que si las diferencian.

Si devuelve un valor, se hace con la sentencia "return", igual que Java.

Se puede indicar el tipo que devuelve la función, de la siguiente forma:

```
function nombreFuncion(<tipo> $param) : <tipo_salida> { ... }
```

Ejemplo:

```
function sumar(int $num1, int $num1) : int {
        return num1 + num2;
}
```

PHP – Ámbito de las variables

En PHP, el ámbito de una variable, según donde se definan, es:

- Local, si se define dentro de una función
- Global, si se define fuera de una función, en el cuerpo del fichero

```
<?php
# Esto es un fichero PHP
$nombreUsuario = "José Luis";                    // Variable global

function mostrarNombre(string $nombre) {
        $mensaje = "hola" . $nombre; # mensaje es local
        echo $nombreUsuario; # Esto falla, aunque sea variable global
}
?>
```

PHP – Ámbito de las variables

Para acceder a variables globales desde una función, se puede:

- Usar la palabra reservada "global"
- Usar el array superglobal $GLOBALS

Ejemplo:

```
<?php
$nombreUsuario = 'José Luis';
function mostrarNombreUsuario() {
        global $nombreUsuario;       # Declarar global en ámbito local
        echo $nombreUsuario;         # Usar global
        echo $GLOBALS['nombreUsuario']  # Forma alternativa
}
?>
```

PHP – Objetos

El uso de objetos es bastante similar a Java. Suponiendo esta clase:

```
<?php
class Persona {
	public $nombre;
	public $edad;

	public function saludar() {
		echo "Hola, mi nombre es " . $this->nombre;
	}
}
?>
```

Como se ve, tiene atributos públicos. Independientemente de que esta práctica sea o no la correcta, trabajaremos con esta clase.

PHP – Objetos

Instanciación de objetos:

```
$persona = new Persona();
```

Acceso a miembros: se usa flecha (->), en lugar de punto (.)

```
$persona->nombre = "Juan";          // Asignar valores
$persona->edad = 30;

echo $persona->nombre;              // Acceder a valores
echo $persona->edad;
```

Ejecución de métodos se usa también la flecha (->):

```
$persona->saludar();    // Llamada a un método
```

PHP – Objetos genéricos – Clase stdClass

Son unos objetos similares a los objetos en JavaScript, que funcionan de forma dinámica, y a los que se pueden añadir atributos de forma dinámica.

Para crear estos objetos se usa la clase stdClass, y se pueden añadir atributos en forma dinámica.

```
$usuario = new stdClass();
$usuario->nombre = 'José Luis';          # Crea el atributo y da valor
$usuario->email = 'jl@example.com';
```

Luego se puede acceder a los atributos sin problema

```
echo $usuario->nombre;
echo $usuario->email;
```