DAW2- Desarrollo Web en Entorno Servidor

UD-3 Lenguaje PHP

Sesión 2

Repaso sesión 1

- Introducción Lenguaje PHP
- Sintaxis básica
  - Abrir Scripts
  - Comentarios
  - Variables
  - Constantes
  - Estructuras de control
  - Operadores
- Funciones auxiliares o nativas

Índice

Arrays

- Un array en PHP es un tipo especial de datos que representa los llamados mapas ordenados de datos.
- Los valores que almacenamos en un array no tienen por qué ser del mismo tipo como ocurre en otros lenguajes de programación.
- Todos los arrays empiezan en el índice 0.
- PHP soporta tanto arrays escalares (índice numérico), como arrays asociativos (índice por clave).
- Para acceder a los elementos de un array se utilizan los corchetes [], dentro de los cuales se indicará un índice o clave de localización.

Arrays: Tipos

- Escalares
  - Los arrays escalares son aquellos en los que para acceder a los elementos utilizamos un índice que representa la posición del valor dentro del array comenzando desde el índice 0.
    - array(valor1, valor2, valor3, ...);
    - [Valor1, valor2, valor3, ...];

Arrays: Tipos

```php
<?php
// Array escalar de valores numéricos  utilizando la notación de corchetes
   $variable = [10, 20,  30, 40, 50];
   echo $variable[0]; -> 10
   echo $variable[3]; -> 40

// Array escalar de valores cadena  utilizando la notación de corchetes
    $variable = ['Enero', 'Febrero', 'Marzo', 'Abril'];
    echo $variable[0]; -> 'Enero'
    echo $variable[2]; -> 'Marzo'

// Array escalar de valores combinados  utilizando la notación de corchetes
    $variable = ['Enero', 10, 'Marzo',  20, 'Abril'];
    echo $variable[0]; -> 'Enero'
    echo $variable[3]; -> 20
```

Arrays: Tipos

- Asociativos
  - Para acceder a los elementos del array utilizamos la clave asociada con el elemento, donde este toma un cierto número de parejas utilizando la sintaxis clave => valor como argumentos.
  - Siempre deben usarse comillas alrededor de un índice de array tipo string literal. Por ejemplo, $array[‘Nombre'] es correcto, mientras que $array[Nombre] no lo es.
    - array(clv1=>valor2,clv2=>valor2,clv3=>valor3,...);
    - [clv1=>valor1,clv2=>valor2,clv3=>valor3,...];

Arrays: Tipos

```php
<?php
// Array asociativo con claves string y valores  numéricos
$Notas = ['Luis'=>6, 'Carmen'=>6, 'Pedro'=>3, 'Rosa'=>8);
echo $Notas['Pedro']; -> 3
$indice='Luis';
echo $Notas[$indice]; -> 6 // Accedemos al elemento por medio de una variable

// Array asociativo con claves string y de  valores string
$Capitales= ['España'=>'Madrid', 'Francia'=>'Paris', 'Italia'=>'Roma',
'Alemania'=>'Berlin');
echo $Capitales['España']; -> 'Madrid'
echo $Capitales[0]; -> Error de índice

// Array asociativo de claves y valores  numéricos y cadena
$Valores= ['Peso'=>65, 10=>'Diez', 'Altura'=>1.75, 20=>'Nombre');
echo $Valores['Altura']; -> 1.75
echo $Valores[20]; -> 'Nombre'
```

Arrays: Tipos

- Multidimensional
  - Un array multidimensional es aquel cuyos valores son otros arrays. Para acceder a sus elementos se tienen que indicar los índices de cada una de sus dimensiones, utilizando tantos pares de corchetes como dimensiones se definan en el array: [][] -> 2 dimensiones, [][][] -> 3 dimensiones, etc….
  - Podemos definirlos con arrays escalares y asociativos o una combinación de ambos.

Arrays: Definición

- En PHP no hace falta definir el array antes de utilizarlo. Cuando se definen elementos de un array, PHP reconoce automáticamente que se trata de un array sin necesidad de declaración previa.
- Podemos definir arrays utilizando el constructor del lenguaje array() o utilizando la notación de corchetes [] también llamada sintaxis corta de array().

Arrays: Definición

- Con array

array(Valor1, Valor2, Valor3, ...);
array(); -> Array vacío

```php
<?php
$numeros = array(10, 20, 30, 40, 50);
$nombres = array('Luis', 'Juan', 'Pedro');
$vacio = array();
```

Arrays: Definición

- Con corchetes (la que se recomienda hoy día)

  [Valor1, Valor2, Valor3, ...];
  [] -> Array vacío

```php
<?php
$numeros = [10, 20, 30, 40, 50];
$nombres = ['Luis', 'Juan', 'Pedro'];
$vacio = [];
```

Arrays: Asignar valores

- Para asignar valores a los elementos de un array, utilizaremos la sintaxis:
  - $NombreArray[IndiceClave]=ValorAsignado;

```php
<?php
$elementos[10]=100.25;
$elementos[11]=50;
$elementos['Precio']=9.99;
```

- También se pueden asignar valores a un array sin especificar el índice y PHP los coloca automáticamente al final de los elementos existentes.
  - $NombreArray[]=ValorAsignado;

```php
<?php
$elementos[]=100.25;
$elementos[]=50;
```

Arrays: Eliminar valores

- Para eliminar algún elemento del array, utilizaremos la función ***unset()*** indicando como argumento la variable de tipo array y el índice o clave del elemento que deseamos eliminar.
- Una vez eliminado el elemento, PHP no reorganiza los índices del resto de los elementos, es decir; si eliminamos el elemento [3], el índice 3 del array habrá desaparecido y si intentamos acceder de nuevo al elemento utilizando ese número de índice se producirá un error de ejecución.

```php
<?php
$numeros=[10,20,30,40];
echo $numeros[2];   // Mostramos el elemento con índice 2 (30)
unset($numeros[2]); // Eliminamos el elemento
echo $numeros[3];   // Mostramos el elemento con índice 3 (40)

// Si intentamos mostrar el elemento con índice 2, se producirá un error de ejecución
echo $numeros[2]; // Notice: Undefined offset: 2
```

Arrays: Eliminar valores

- Para mantener el orden de los índices después de eliminar un elemento, tendremos que reasignar los valores al array con la función ***array_values()***.

```php
<?php
// Definimos un array de 4 elementos
$numeros=[10,20,30,40];
unset($numeros[2]); // Eliminamos el elemento con índice 2 (30)
$numeros= array_values($numeros); // Se reasigna los valores al array
echo $numeros[2]; // Mostramos el elemento con índice 2 (40)
```

Arrays: Unión

- El operador de unión + devuelve el array del lado derecho añadido al array del lado izquierdo; para las claves que existan en ambos arrays, serán utilizados los elementos del array de la izquierda y serán ignorados los elementos correspondientes del array de la derecha.

```php
<?php
$array1 = [10, 20, 30];
$array2 = [40, 50];
$array = $array1 + $array2; // Resultado [10, 20, 30]
$array = $array2 + $array1; // Resultado [40, 50, 30]
```

- La función array_merge() combina los elementos de uno o más arrays juntándolos de modo que los valores de uno se anexan al final del anterior, devolviendo el array resultante.

```php
<?php
$array1 = [10, 20, 30];
$array2 = [40, 50];
$union = array_merge($array1,$array2); // Resultado [10, 20, 30, 40, 50]
```

Arrays: Funciones auxiliares

- Contar los elementos de un array
  - *count()*
- Mostrar su estructura y contenido
  - *print_r()*
- Buscar un valor
  - *in_array()*
  - *array_search()*
- Buscar un índice o clave
  - *array_key_exists()*

Funciones

- Cada función representa una unidad de procesamiento reutilizable que puede recibir uno o varios parámetros y devolver un valor.
- La escritura de funciones permite estructurar el código desacoplando de manera lógica las funcionalidades desarrolladas. Se recomienda, para una buena legibilidad y una buena mantenibilidad, limitar la longitud de las funciones.
- Debemos limitar la responsabilidad de las funciones: cada una debería realizar un tipo de tarea únicamente.
- Cuando nos referimos a función es sinónimo de método o procedimiento.

Funciones

- En PHP para definir una función se utiliza la palabra reservada ***function*** seguida del nombre de la función y paréntesis donde se indican los argumentos de trabajo y se encierra las líneas de código entre llaves {} y un valor de retorno (usando ***return***) que es opcional.

```
function NombreFuncion([Argumentos]) {
  Líneas de código
  [return [ValorRetorno]]
}
```

- Cualquier código PHP válido puede aparecer dentro de una función, incluso otras funciones.
- No es necesario definir una función antes de que sea referenciada, excepto cuando esta esté condicionalmente definida dentro de sentencias condicionales como el *if()* o hayan sido definidas dentro de otras funciones.

Funciones

- Un nombre de función válido comienza con una **letra o guión bajo**, seguido de cualquier número de letras, números o guiones bajos.
- Los nombres de funciones son Case-insensitive, es decir **no se hace la distinción entre mayúsculas y minúsculas**.
- Por convención siempre las escribiremos en minúsculas y comenzarán también en minúsculas. Si una función tiene más de una palabra, cada palabra a excepción de la primera debe de comenzar en mayúsculas.

```php
<?php

function bienvenida() {
      echo '<p>Bienvenido a mi web</p>';
}

bienvenida();
BIENVENIDA(); // Es correcto ya que los nombres de función son case-insensitive.
# Daría como resultado dos párrafos HTML con el texto Bienvenido a mi web.
```

Funciones: Argumentos

- Podemos pasar cualquier información a las funciones mediante la lista de argumentos delimitadas por comas. Los argumentos son evaluados de izquierda a derecha, lo que implica que se tienen que enviar a la función en el mismo orden en el que hayan sido definidos.

```
function nombreFuncion($Argumento1, $Argumento2, ...) {
  Líneas de código
}
```

- El paso de argumentos se realiza de forma predeterminada por valor, es decir se envía una copia del valor. También admite el paso de argumentos por referencia anteponiendo el símbolo **&** al nombre del argumento y listas de argumentos de longitud variable.

```
function nombreFuncion($Argumento1, &$Argumento2, ...) {
  Líneas de código
}
```

Funciones: Argumentos

- Si deseamos declarar **valores por defecto o predeterminados** para los argumentos, los definiremos realizando la asignación del valor. Este valor será el que se utilice en el caso de no recibirse el argumento durante la llamada a la función. Cuando se emplean argumentos predeterminados, tendrán que ser declarados a la derecha de los argumentos no predeterminados, es decir; tendrán que ser siempre los últimos de la lista de argumentos de la función.

```
function nombreFuncion($Argumento1, $Argumento2=Valor, $Argumento3=Valor,  ..) {
  Líneas de código
}
```

- PHP admite el uso de arrays para declarar argumentos predeterminados en nuestras funciones, lo que nos permite en un solo argumento enviar múltiples valores.

Funciones: Devolver valores

- Cuando una función tiene que devolver un valor de retorno utiliza la sentencia opcional ‘return’ seguida del valor que desea retornar. Se puede devolver cualquier tipo de datos incluso arrays y objetos.
- Esta sentencia provoca la finalización de la ejecución de las líneas de código de la función y devuelve el control a la instrucción que la referenció. Si se omite return dentro de la función el valor devuelto será siempre NULL.

```php
<?php
function es_par($numero) {
     if($numero%2==0){
         return true;
     }else{
         return false;
     }
}
```

Funciones: Funciones variables

- PHP admite el concepto especial de funciones variables. Esto significa que si un nombre de variable tiene paréntesis anexos a él, PHP buscará una función con el mismo nombre que lo evaluado por la variable, e intentará ejecutarla.

```php
<?php
function cuadrado($numero) {
	return ($numero*$numero);
}
function doble($numero) {
	return ($numero*=2);
}
// Asignamos a la variable el nombre de la función que deseamos que represente
$func='doble';
// Invocamos a la función añadiendo paréntesis al nombre de la variable
echo $func(25); // Muestra 50
$func='cuadrado';
echo $func(8); // Muestra 64
```

Inclusión de ficheros

- Si tenemos funciones que nos pueden servir para los scripts de nuestro aplicativo web, lo que tenemos que hacer es agruparlas en un único archivo php, formando lo que se llama una librería.
- Cada vez que necesitemos utilizar alguna de las funciones, sólo tendremos que incluir la librería al principio de nuestro script para poder utilizarlas.
- Para poder incluir un archivo de librería en nuestros scripts, tendremos utilizar alguna de las siguientes sentencias de PHP:
  - *include, include_once*
  - *require, require_once*

Inclusión de ficheros

- Las sentencias ***include*** y ***require***, incluyen en nuestro código el contenido del archivo que se le pase como argumento y lo evalúan.
  - *include(RutaArchivoPHP)*
  - *require(RutaArchivoPHP)*
- Los archivos se incluirán en nuestro script con base a la ruta de acceso dada, o si no se indica ruta de acceso se buscarán en el directorio actual o en la lista de directorios en la directiva include_path de PHP.
- Cuando se incluye un archivo, el código que contiene hereda el ámbito de las variables de la línea en la cual ocurre la inclusión.

Inclusión de ficheros

- La principal diferencia entre ***include*** y ***requiere***, se produce en el caso de no encontrarse el archivo a incluir. Mientras que ***include*** simplemente generará un mensaje de aviso y la ejecución continua, con ***requiere*** se genera un error fatal que provoca la interrupción del código.
- include_once y require_once
    - De funcionamiento análogo a *include* y *require*, siendo la única diferencia: si el código del fichero ya ha sido incluido, no se volverá a incluir, e include_once y require_once devolverán TRUE. Como su nombre indica, el fichero será incluido solamente una vez.

Manejo de ficheros

- El lenguaje PHP dispone de una gran cantidad de funciones para realizar todo tipo de operaciones con archivos y carpetas, tanto básicas (como crear archivos y carpetas, modificar, eliminar...) como otras más avanzadas (para obtener y asignar permisos, crear enlaces simbólicos, etc.).
    - https://www.php.net/manual/es/book.filesystem.php

Manejo de ficheros

- Comprobar si existe un archivo o directorio
  - Si necesitamos averiguar si existe un archivo o carpeta utilizaremos la función de PHP file_exists(), la cual nos devolverá true si existe false en caso contrario:

```php
<?php
	if( file_exists("datos.txt") == true)
		echo "<p>El archivo existe</p>";
	else
		echo "<p>El archivo no se ha encontrado</p>";

	if( file_exists("miCarpeta") == true )
		echo "<p>El directorio existe</p>";
	else
		echo "<p>El directorio no existe</p>";
```

Manejo de ficheros

- Abrir el fichero:

```php
<?php

// Abrir el archivo en modo lectura:
$archivo = fopen("datos.txt","r");
```

- Cerrar el fichero:

```php
<?php
   // Cerrar el archivo:
   fclose($archivo);
```

Manejo de ficheros: Lectura

- Para leer fichero línea a línea en PHP podemos utilizar la función ***fgets()***

```php
<?php
$file = fopen("archivo.txt", "r") or exit("Unable to open file!");
//Output a line of the file until the end is reached
while(!feof($file))
{
echo fgets($file). "<br />";
}
fclose($file);
```

Manejo de ficheros: Escritura

- Para escribir un archivo de texto en PHP podemos utilizar la función ***fwrite()*** o ***fputs()***.

```php
<?php

$file = fopen("archivo.txt", "w");
fwrite($file, "Esto es una nueva línea de texto" . PHP_EOL);
fwrite($file, "Otra más" . PHP_EOL);
fclose($file);
```

Recursos

- http://minubeinformatica.com/cursos/programacion-en-php/arrays-mapas-de-datos
- https://programadorwebvalencia.com/cursos/php/arrays/
- https://diego.com.es/arrays-en-php
- https://codesamplez.com/programming/php-arrays-tutorial
- https://www.baulphp.com/como-usar-include-y-require-en-php-ejemplos/
- https://desarrolloweb.com/articulos/312.php
- https://desarrolloweb.com/articulos/parametros-funciones-php.html
- https://informaticapc.com/tutorial-php/crear-archivos.php