DWES

Introducción a PHP

2º DAW

Índice

1. Introducción

- Es un **lenguaje de programación del lado del servidor,** es decir, se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente, por eso **necesitamos instalar un servidor en local para poder probar los scripts.**
- Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución del código PHP.
- Se trata de uno de los lenguajes del lado servidor más populares y Muchas aplicaciones web están construidas con PHP. Ejs.: Wordpress, Joomla, Drupal, osCommerce, Prestashop, phpBB, SMF, Moodle,etc.

PHP -> acrónimo recursivo de PHP HypertextPreprocessor.

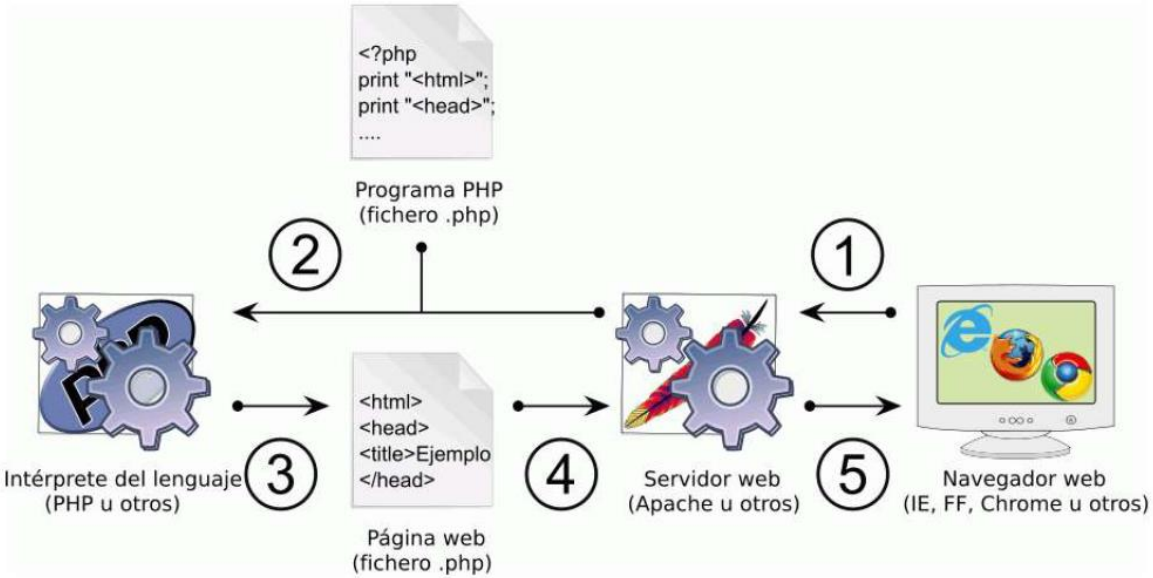
Características

- ✓ Es un lenguaje interpretado
- ✓ Basado en C y Perl, por tanto similar a Java
- ✓ Menos estricto semánticamente que Java
- ✓ Es un lenguaje Web

Ventajas

- ✓ Multiplataforma (Windows, Linux, Mac…)
- ✓ Abierto y gratuito.
- ✓ Muy utilizado lo que proporciona una amplia red de consultas y extensiones que amplían la capacidad del lenguaje.

1. Introducción



1. El cliente hace una petición al servidor solicitando una página.
2. El servidor recibe el mensaje, y al ver que se trata de un archivo con código PHP solicita al intérprete de PHP (otro programa que se ejecuta en el servidor) que le envíe el archivo.
3. El intérprete lee el contenido del archivo y ejecuta sus instrucciones (pueden incluir consultas a BD, generar dinámicamente contenido, etc.)
4. Envía el archivo con el resultado de la ejecución al servidor.
5. El servidor envía el archivo resultante al cliente que lo había solicitado y este se visualiza en el navegador.

2. Preparación del entorno

- **Ejecución de un script PHP**

  - ✓ Los archivos deben tener la extensión ".php“.

  - ✓ Los archivos deben desplegarse en el directorio raíz de archivos del sitio Web: "C:/xampp/htdocs“ (Este directorio puede cambiarse mediante la directiva DocumentRoot, que se encuentra en el archivo de configuración C:\xampp\apache\conf\httpd.conf).

  - ✓ El servidor Web (Apache) con el módulo para PHP debe estar iniciado.

  - ✓ En el cliente web, es decir, en la barra de direcciones del navegador, debemos escribir la ruta al archivo mediante el protocolo http.
    - Para pruebas locales http://localhost/manual/prueba.php
    - Para pruebas en remoto se debe indicar la dirección IP o el nombre de dominio: http://www.juanmacr.es/manual/prueba.php

- **Sintaxis**

  - ✓ El código PHP puede ir escrito dentro de un documento HTML entre las etiquetas <?php…. ?>.
  - ✓ Cada instrucción ha de finalizar con un “;”

2. Preparación del entorno

- El código PHP puede ir escrito dentro de un documento HTML entre las etiquetas <?php…. ?>.
- Cada instrucción ha de finalizar con un “;”

```
<doctype html>
<html>
 <head><title> Mi primer php </title></head>
 <body>
   <?php
       echo "Mi primer php";  //Muestra el mensaje por pantalla
   ?>
 </body>
</html>
```

Los comentarios son fragmentos de código que no se ejecutan ni visualizan. Son ignorados por el ordenador.

- Comentarios de una línea usando //(al estilo de C)
- Comentarios de una línea usando #(al estilo de Linux)
- Comentarios de varias líneas /* … */(al estilo de C)

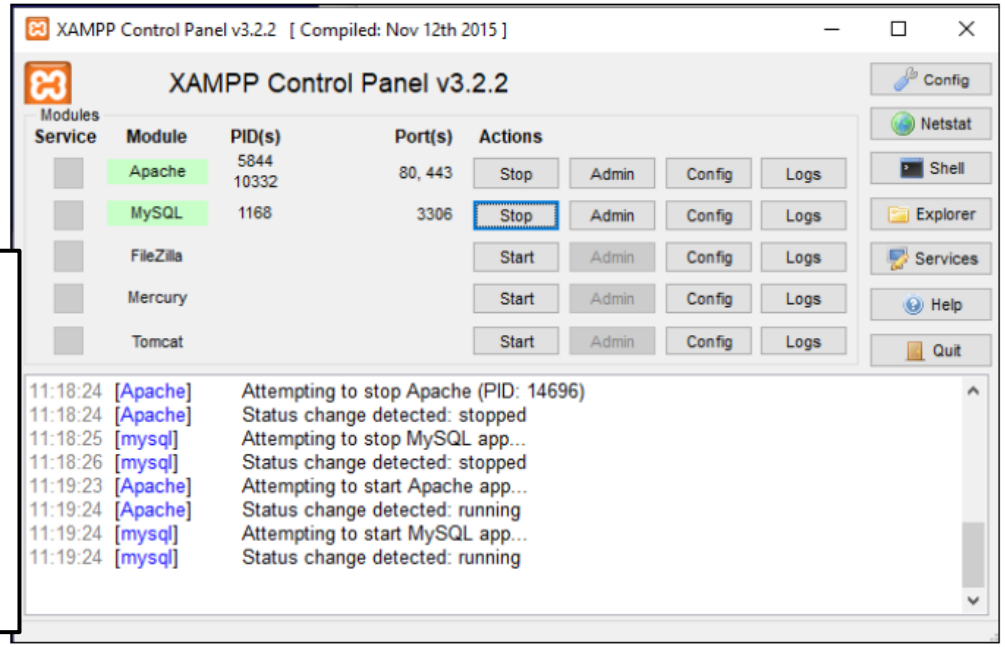
2. Preparación del entorno

XAMMP es la herramienta que nos va a permitir ejecutar programas en entorno servidor.
Incluye:

- ✓ Un servidor Apache.
- ✓ Un gestor de BBDD MySQL.
- ✓ El intérprete de PHP.

Para arrancar el servidor y el gestor de BBDD, debemos arrancar “XamppControl Panel”, que corresponde con el archivo “xampp-control.exe” que se encuentra en el directorio raíz de XAMPP.

- Para iniciar el servidor Apache pulsaremos el botón “Start” que figura a su lado.
- Para iniciar el gestor MySQL pulsaremos el botón “Start” que aparece a su lado.
- Si todo ha ido bien aparecerá en la parte inferior de la ventana el status “running”.



2. Preparación del entorno

- Puedes comprobar que funciona correctamente pulsando en los botones “Admin” tanto de Apache como de MySQL.
- Si todo es correcto, se mostrarán en el navegador la página de inicio del servidor Apache (observa la URL), y la herramienta PHPMyAdminque permite gestionar BBDD.

2. Preparación del entorno

Vamos a probar nuestro primer script, para ello, utiliza un editor y realiza los siguientes pasos:

1. Asegura que tienes activo Apache (paquete XAMPP)
2. Crea un nuevo documento html que incluya fragmento de código php. Puedes utilizar un echo para mostrar un mensaje o la función phpinfo(); para ver que el interprete y la instalación funciona correctamente.Esta función de PHP muestra por pantalla todas las opciones de configuración de PHP que se encuentran en el archivo php.ini
3. Coloca el fichero en la carpeta "C:/xampp/htdocs“
4. Prueba el correcto funcionamiento desde el navegador (recuerda que estas trabajando en local)

Tarea: Busca información sobre diferentes editores de código (Netbeans, Dreamweaver, braket, Visual Studio, Notepad++). Identifica ventajas e inconvenientes teniendo en cuenta el lenguaje de programación a trabajar, ¿con cual te quedas y por qué?

3. Lenguaje PHP

- **Variables**

Las variables se usan para guardar información en memoria.

- Los nombres de variables ha de comenzar con el símbolo $. Tras él, puede ir un guion bajo o una letra, no un número. A partir del tercer carácter puede ser un número, letra o guion bajo.
- PHP es un lenguaje no tipado. Automáticamente le asigna el tipo al asignarle un valor.
- Además podemos cambiar de tipo de dato en cualquier momento asignándole cualquier valor a una variable durante el programa.
- Es un lenguaje case-sensitive, es decir distingue entre mayúsculas y minúsculas, por lo que las variables $numero y $Numero son variables diferentes.

Las cadenas de caracteres se pueden declarar:

- Con comillas simples.
- Con comillas dobles.
- Ambos métodos tienen un comportamiento similar. Si embargo, si queremos mostrar dentro de una cadena de caracteres el contenido de una variable, debemos utilizar comillas dobles. Si se usaran comillas simples, se mostraría el nombre de la variable, no su contenido.

3. Lenguaje PHP

- **Variables**

```php
<?php
        $nombre1 = "Juanma";        // variable de tipo string
        $nombre2 = "Joserra";       // variable de tipo string
        $nivel = 24;                // variable de tipo integer
        $profe = TRUE;              // variable de tipo boolean
        $_sueldo = 'poco';          // variable de tipo string
        echo  "$nombre1, $nombre2, $nivel, $profe, $_sueldo";
?>
```

```php
<?php
    $dato = "IMPRIMIBLE";
    $texto_simple = 'No puede imprimir la variable $dato';
    $texto_doble = "Imprime la variable dato como $dato";
    echo ("$texto_simple<br>");
    echo ("$texto_doble");
?>
```

3. Lenguaje PHP

❖ **Variables**

En PHP las variables pueden ser declaradas en cualquier lugar del código.

```php
<?php
 $x=10;   $y=20;          // ámbito global

 function ambito() {
   $x=1;   $y=2;          // ámbito local a la función
   echo "Variables locales a la función:  <br>";
   echo "x = $x  <br>";    echo "y = $y  <br>";
   global $x, $y;
   echo "Variables globales:  <br>";
   echo "x = $x  <br>";  echo "y = $y  <br>";
 }
 ambito();
?>
```

- El ámbito de una variable es la parte del código donde puede ser usada, PHP tiene los siguientes ámbitos para las variables:
- **Local:** las variables declaradas dentro de una función, tiene ámbito local y solo puede ser usada dentro de dicha función (incluso aunque tenga el mismo nombre que una variable global)
- **Global:** las variables declaradas fuera de toda función tienen un ámbito global y solo puede ser usadas fuera de toda función.
- Una variable global puede utilizarse dentro de una función si en la función se antepone la palabra global al nombre de la variable.

3. Lenguaje PHP

❖ **Variables**

Las constantes son datos cuyo valor no varía durante la ejecución de un programa:

- Para definir una constante en PHP se utiliza la función define().
- Por ejemplo, define(“ROJO”,”#FF0000”);
- Los nombres de las constantes no van precedidas del símbolo $.
- Por convención se suelen escribir en mayúsculas.
- Solo pueden tener valores de tipo string, boolean, integery float.
- Para mostrar su contenido: echo ROJO

```
<?php
    define ("Constante1", "Hola");          // constante declarada mediante define()
    const Constante2 = "mundo";             // constante declarada mediante const
    echo  Constante1, " " ,  Constante2;

?>
```

3. Lenguaje PHP

❖ **Variables**

Se trata de un conjunto de variables predefinidas y accesibles desde cualquier ámbito (funciones, clases o archivos).

Los tipos de variables superglobales en PHP son:

- **$GLOBALS:** contiene todas las variables globales definidas en el script.
- **$_SERVER:** contiene las variables del servidor Web (cabeceras, rutas, etc.) .
- **$_REQUEST:** contiene los datos enviados en un formulario HTML.
- **$_POST:** contiene los datos enviados en un formulario HTML con method=”post”.
- **$_GET :** contiene los datos enviados en un formulario HTML con method=”get”.
- **$_FILES:** contiene variables proporcionadas por medio de ficheros.
- **$_ENV:** contiene las variables proporcionadas por el entorno.
- **$_COOKIE:** contiene las variables proporcionadas por cookies.
- **$_SESSION:** contiene las variables registradas en la sesión del script.

3. Lenguaje PHP

❖ **Operadores**

Son muy parecidos a los de otros lenguajes de programación:

- Asignación: se utiliza el operador =; $num=7;
- Aritméticos:
- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Módulo (%): resto de la división entera
- Unario (-): el opuesto, por ejemplo-$a;

$a += $b;(equivale a $a = $a + $b)
$a -= $b; (equivale a $a = $a -$b)
$a *= $b;(equivale a $a = $a * $b)
$a /= $b;(equivale a $a = $a / $b)
$a.= $b;(equivale a $a = $a . $b) **(concatenación)**
$a %= $b; (equivale a $a = $a % $b

- Incremento:
  - ✓ $b = $a++; (asigna el valor de $a a$b y después incrementa el valor de a una unidad).
  - ✓ $b = ++$a; (incrementa el valor de $a en una unidad y después lo asigna a b).

- Decremento:
  - ✓ $b = $a--; (asigna el valor de $a a$b y después decrementa el valor de a una unidad).
  - ✓ $b = --$a; (decrementa el valor de $a en una unidad y después lo asigna a b).

3. Lenguaje PHP

- **Operadores**

De comparación:

- ✓ $a==$b(devuelve true si el valor de a y b es igual)
- ✓ $a===$b (devuelve true si el valor de a y b es igual, y además son del mismo tipo.
- ✓ $a != $b o también $a <> $b (devuelve true si son distintos).
- ✓ a < $b(devuelve true si $a es menor que $b)
- ✓ $a > $b(devuelve true si $a es mayor que $b)
- ✓ $a <= $b(devuelve true si $a es menor o igual que $b)
- ✓ $a >= $b(devuelve true si $a es mayor o igual que $b)

Lógicos:

- ✓ $a and $b o también$a && $b (devuelve true si $a y $b son ciertos).
- ✓ $a or$b--o también$a || $b (devuelve true si $a o $b son ciertos).
- ✓ !$a (devuelve true si $a es falso). Negación
- ✓ $a xor $b (devuelve true si uno de los dos es cierto, pero no ambos).

Llaves: el operador { } sirve para indicar al intérprete cual es la parte que debe procesar, dentro de una cadena entre comillas dobles. Se puede utilizar para visualizar matrices, funciones, métodos de objetos, etc. dentro de cadenas.

3. Lenguaje PHP

```
<body>
    <?php
    /* Fíjate cómo se concatenan los elementos para mostrar un resultados */
        $cadena="5"; //esto es una cadena
        $entero=3; //esto es un entero
        echo "1) el valor resultante es <br>";
        echo $cadena+$entero;
        echo "<br>";
        echo "2) el valor resultante es $cadena+$entero<br>";
        echo "3) el valor resultante es ($cadena+$entero)<br>";
        echo "4) el valor resultante es ".($cadena+$entero)."<br/>";
        echo "5) el valor resultante es ".$cadena."+".$entero."=".($cadena+$entero);

        // Al asignarle el valor 7, la variable es de tipo “entero”
        $mi_variable = 7;
        // Si le cambiamos el contenido
        $mi_variable = "siete";
        // La variable puede cambiar de tipo
        // En este caso pasa a ser de tipo “ cadena”
        echo "<br><br/>";
        $mi_entero = 3;
        $mi_real = 2.3;
        $resultado = $mi_entero + $mi_real;
        // La variable $resultado es de tipo real
        echo "el resultado es $resultado\n";
        /* la secuencia \n inserta un salto de línea en el código que se genera, no en la página
        web
         * así resulta más legible
         *  */
        echo "<br/>";
        $mi_entero = 3;
        $mi_real = 2.3;
        $resultado =$mi_entero +(int)$mi_real;
        echo "el último resultado (convertido a entero con operador de casting) es $resultado \n";

        // La variable $mi_real se convierte a entero (valor 2) antes de sumarse.
        // La variable $resultado es de tipo entero (valor 5)
    ?>
</body>
```

3. Lenguaje PHP

- **Estructuras de control**

  - **Condicionales: if – elseif -else**

```php
<?php
  $nota = 5;
  if ($nota<5) {
    echo "Vuelve a intentarlo";
  } elseif ($nota==5) {
    echo "Uff";
  } else {
    echo "Bien hecho!";
  }
?>
```

  - **switch:**

```php
<?php
  $genero = 'F';
  switch ($genero) {
    case 'M':  echo 'Masculino';
                 break;
    case 'F':  echo 'Femenino';
                 break;
    default:  echo "Neutro";
  }
?>
```

3. Lenguaje PHP

- **Estructuras de control**

• Bucles: while

```php
<?php
  $i = 0;
  while  ($i<10)  {
    echo "contador = $i";
    $i++;
  }
?>
```

• do while

```php
<?php
  $i = 0;
  do {
    echo "contador = $i";
    $i++;
  }  while ($i<10);
?>
```

3. Lenguaje PHP

❖ **Estructuras de control**

- **Bucles: for**

```
<?php
  for  ($i=0; $i<10, $i++)   {
    echo "contador = $i";
  }
?>
```

- **foreach** (lo veremos más adelante)

```
<?php
  $paises = array ('Italia', 'España', 'Francia');
  foreach  ($paises as $pais)   {
    echo "$pais <br>";
  }
?>
```

3. Lenguaje PHP

- **Funciones**

Una función es un conjunto de instrucciones que realizan una tarea determinada.

- La forma de definir funciones de usuario en PHP es muy similar a la de otros lenguajes:

  Funtionnombre_función($parámetro1, $parámetro2, …){
  Instrucciones …
  }

- Las funciones pueden devolver valores (usando la palabra reservada return)o no, también pueden recibir parámetros o no.

3. Lenguaje PHP

- **Funciones**

```php
function duplicar($var){
    $var *=2;
}
$var =5;

echo "<h2>ejemplo ámbilo local de las variables</h2>";
echo "el valor de la variable ANTES DE llamar a la función duplicar es: $var </br>";
duplicar($var);
echo "el valor de la variable DESPUÉS de llamar a la función duplicar es: $var </br>";
echo"</br>";
echo"<hr></br>";
```

```php
//devolviendo un resultado
function duplicar2($valor){
    $resultado = $valor * 2;
    return ($resultado);
}
echo "<h2>llamar a una función devolviendo un resultado </h2>";
$var=5;
echo "el valor de la variable ANTES DE llamar a la función duplicar2 es: $var </br>";
$var = duplicar2($var);
echo "el valor de la variable DESPUÉS de llamar a la función duplicar2 es: $var </br>";
```

3. Lenguaje PHP

- **Funciones**

```php
echo "<h2>variables no estáticas</h2>";
function contador(){
    // Cada vez que se ejecuta la función, se incrementa el valor de $cont
    $cont=0;
    $cont++;
    echo "CONTADOR: $cont</br>";
}
echo"</br>";
echo"</br>";

for($i=0;$i<10;$i++)
    contador();
```

variables no estáticas

CONTADOR: 1
CONTADOR: 1
CONTADOR: 1
CONTADOR: 1
CONTADOR: 1
CONTADOR: 1
CONTADOR: 1
CONTADOR: 1
CONTADOR: 1
CONTADOR: 1

```php
echo "<h2>variables estáticas</h2>";
    //static
    function contador_estatico(){
        // Cada vez que se ejecuta la función, se incrementa el valor de $a
        static $cont=0;
        $cont++;
        echo "CONTADOR: $cont</br>";
    }


    echo"</br>";
    echo"</br>";

    for($i=0;$i<10;$i++)
        contador_estatico();
```

variables estáticas

CONTADOR: 1
CONTADOR: 2
CONTADOR: 3
CONTADOR: 4
CONTADOR: 5
CONTADOR: 6
CONTADOR: 7
CONTADOR: 8
CONTADOR: 9
CONTADOR: 10

3. Lenguaje PHP

- **include() y require()**

A medida que los proyectos se hacen más complejos es recomendable separar código en ficheros separados con extensión PHP y utilizar las funciones include(fichero)o require(fichero)para que PHP incorpore su contenido dentro del programa en el que se incluya.

- require()detiene la ejecución del script si no puede incluir el fichero. La función include()da un aviso de error, pero intenta continuar con la ejecución del programa.
- Existen dos variantes require_once(fichero)e include_once(fichero)que se aseguran de que el fichero pasado como parámetro solo se incluye una vez en el código evitando así errores que se puedan producir si incluimos más de una vez un mismo fichero (por ejemplo la definiciones de funciones repetidas).

```
<?php
   require 'header.php';
   include 'menu.html';
?>
```

3. Lenguaje PHP

- **Arrays**

  Un array es una colección de valores que se identifican por un único nombre.

  En PHP hay dos tipos de arrays:

  - **Arrays indexados:** el índice que se utiliza para acceder a los elementos del arrayes un número que indica la posición dentro del array. El índice del primer elemento del array es el cero. $miArray[0] = “Hola”;

  - **Arrays asociativos:** utilizan como clave para acceder a los elementos del array una cadena de caracteres. $miArray[“color”] = “rojo”; Para mostrar el contenido completo de un array por pantalla se utilizan las funciones print_r()o var_dump().

- **Formas de crear arrays:**

  - ✓ Asignación directa
  - ✓ Con la función array()
  - ✓ **En PHP no es necesario indicar el tamaño del array al crearlo ni especificar que es un array.**

3. Lenguaje PHP

- **Arrays**
  - **Formas de crear arrays:**

    Asignación directa: $miArray[15]=128;

    - ✓ Crea un array solo con el elemento posición 15. Los demás no existen y si se intenta acceder a ellos se produce un error.
    - ✓ Para añadir un elemento al final, no se indica valor entre los corchetes:

      $miArray[]="ÚLTIMO";

    - ✓ Los elementos del array no tienen por qué ser del mismo tipo de dato.
    - ✓ Si definimos un array asociativo hay que indicar entre los corchetes el string que será la clave de acceso a los elementos del array:

      $miArray["color"]="rojo";

3. Lenguaje PHP

- **Arrays**
  - **Formas de crear arrays:**

    Con la función array(): $miArray= array(23, “Pepe”, 12.58, true);

    - ✓ Si definimos un array indexado, SIN INDICAR EL ÍNDICE, los índices se añaden automáticamente empezando desde 0:
    - ✓ El array ahora tiene cuatro elementos en posiciones consecutivas, cada uno de ellos con un tipo de dato diferente.
    - ✓ Se puede crear un array vacío de la siguiente manera: $miArray= array();
    - ✓ También se puede indicar explícitamente el índice al crear el array de la siguiente manera:
      $miArray= array(1=>50, 3=>30,15=>60);
    - ✓ Recuerda que los elementos no definidos no existen, por lo que no se puede acceder a ellos.

3. Lenguaje PHP

- **Arrays**
  - **Formas de crear arrays:**

    Con la función array(): $miArray= array(23, “Pepe”, 12.58, true);

    - ✓ Si definimos un array asociativo, hay que indicar el nombre de la clave, y a continuación, separado por la combinación de caracteres “=>”, el valor del elemento del array al que se accederá mediante esa clave:

      ```
      $miArray= array(“it”=> “Italia”, “es” => “España”, “fr”=> “Francia”);
      Echo $miArray[“es”];
      ```

    - ✓ Un mismo array puede tener índices tanto de tipo numérico como de tipo asociativo.

      ```
      $aCosas= array( 12=>”Madrid”, "color" => "rojo", "importe" => 300, "activo" => true, 3 => 55 );
      ```

    - ✓ En los arrays, si se repite un índice o clave, se sobrescribe el valor que tuviera y solo aparecerá el último valor.
    - ✓ Si se intenta acceder a un elemento de un array asociativo utilizando un índice numérico da error.

3. Lenguaje PHP

- **Arrays**

  - **Recorrer arrays:**

    - Los **arrays indexados** (los que tienen índice numérico), pueden recorrerse de la manera tradicional utilizando **una estructura de tipo bucle**.
    - IMPORTANTE Si el array no tiene definido un elemento en una determinada posición se producirá un error al intentar acceder a él.
    - Una forma de asegurarse de que no nos aparezca un error, es comprobando si existe el elemento del array antes de hacer nada con él.
    - Otra forma es utilizando el **bucle foreach.**

3. Lenguaje PHP

```php
$miArray[0]=45;
$miArray[12]="pepe";
$miArray[]="ÚLTIMO";
print_r($miArray);
echo "<br/><br/>";


echo "<h2>Mostrar el contenido usando un bucle</h2>";
echo "la función count devuelve el nº de elementos que contiene un "
. "array sin tener<br/> en cuenta las posiciones vacías. en este caso el"
        . " array tiene ".count($miArray)," elementos pero, <br/> por ejemplo el elemento "
        . "con posición 1 2, por lo que da error al acceder a uno de ellos<br/><br/>";
for($i=0;$i<count($miArray);$i++){
    echo "elemento posición ".$i." = ".$miArray[$i];
    echo "<br/>";

}
```

En el ejemplo, la función count devuelve un 3, el array contiene 3 elementos, pero no están en posiciones consecutivas. Al intentar acceder a los elementos con posiciones 1 y dos se produce un error.

Mostrar el contenido usando un bucle

la función count devuelve el nº de elementos que contiene un array sin tener
en cuenta las posiciones vacías. en este caso el array tiene 3 elementos pero,
por ejemplo el elemento con posición 1 2, por lo que da error al acceder a uno de ellos

elemento posición 0 = 45

Notice: Undefined offset: 1 in C:\xampp\htdocs\DWES_P_U1\11_arrays_02_recorrer.php on line 25

Call Stack

#	Time	Memory	Function	Location
1	0.2050	391256	{main}()	...\11_arrays_02_recorrer.php:0

elemento posición 1 =

Notice: Undefined offset: 2 in C:\xampp\htdocs\DWES_P_U1\11_arrays_02_recorrer.php on line 25

Call Stack

#	Time	Memory	Function	Location
1	0.2050	391256	{main}()	...\11_arrays_02_recorrer.php:0

elemento posición 2 =

3. Lenguaje PHP

```php
echo "<h2>Mostrar el contenido usando un bucle comprobando si existe</h2>";
echo "el nº de elementos del array no tiene en cuenta los elementos vacíos<br>";
$tam=count($miArray);
echo "el array tiene $tam elementos que son:<br/> ";
print_r($miArray);
echo "<br/><br/>";
for($i=0;$i<$tam;$i++){
    if (isset($miArray[$i])){
        echo "elemento índice ".$i." = ".$miArray[$i].", tam = $tam <br/>";
    }
    else{
        $tam++;
        echo "el elemento índice ".$i." no existe, tam= $tam<br/>";

    }
}
```

el nº de elementos del array no tiene en cuenta los elementos vacíos
el array tiene 3 elementos que son:
Array ([0] => 45 [12] => pepe [13] => ÚLTIMO)

elemento índice 0 = 45, tam = 3
el elemento índice 1 no existe, tam= 4
el elemento índice 2 no existe, tam= 5
el elemento índice 3 no existe, tam= 6
el elemento índice 4 no existe, tam= 7
el elemento índice 5 no existe, tam= 8
el elemento índice 6 no existe, tam= 9
el elemento índice 7 no existe, tam= 10
el elemento índice 8 no existe, tam= 11
el elemento índice 9 no existe, tam= 12
el elemento índice 10 no existe, tam= 13
el elemento índice 11 no existe, tam= 14
elemento índice 12 = pepe, tam = 14
elemento índice 13 = ÚLTIMO, tam = 14

Usando la función isset() comprobamos si el elemento existe.
Además debemos incrementar la variable $tam para asegurarnos de que vamos a llegar hasta el final del array.

3. Lenguaje PHP

Recorrer arrays: foreach

```php
$notas= array("Antonio"=>5,
              "Luisa"=>6.5,
              "Ana"=>9,
              "Gabriel"=>8.5,
              "Eloy"=>6,
              "Jaime"=>7,
              "Berta"=>5);
echo "Tenemos el array asociativo notas que contiene las notas de "
. "los exámenes de alumnos:<br/>";
print_r($notas);
echo "<br/><br/>";
echo "<h2>Recorremos array con foreach con la sintaxix foreach(array as "
. "valor) y mostramos el valor de cada elemento</h2>";
foreach($notas as $nota)
    echo "bucle foreach nota = ".$nota."<br/>";
echo "<h2>Recorremos array con foreach con la sintaxix "
. "foreach(array as clave =>valor) y mostramos el valor de cada elemento</h2>";
    foreach($notas as $nombre=>$nota)
    echo "bucle foreach nombre = ".$nombre.", nota = ".$nota."<br/>";
```

3. Lenguaje PHP

Recorrer arrays: foreach

Tenemos el array asociativo notas que contiene las notas de los exámenes de alumnos:
Array ([Antonio] => 5 [Luisa] => 6.5 [Ana] => 9 [Gabriel] => 8.5 [Eloy] => 6 [Jaime] => 7 [Berta] => 5)

Recorremos array con foreach con la sintaxix foreach(array as valor) y mostramos el valor de cada elemento

bucle foreach nota = 5
bucle foreach nota = 6.5
bucle foreach nota = 9
bucle foreach nota = 8.5
bucle foreach nota = 6
bucle foreach nota = 7
bucle foreach nota = 5

Recorremos array con foreach con la sintaxix foreach(array as clave =>valor) y mostramos el valor de cada elemento

bucle foreach nombre = Antonio, nota = 5
bucle foreach nombre = Luisa, nota = 6.5
bucle foreach nombre = Ana, nota = 9
bucle foreach nombre = Gabriel, nota = 8.5
bucle foreach nombre = Eloy, nota = 6
bucle foreach nombre = Jaime, nota = 7
bucle foreach nombre = Berta, nota = 5

3. Lenguaje PHP

❖ **exit(), die() y header()**

- La función **exit(stringstatus)** imprime un mensaje y finaliza la ejecución del script actual:
  - ✓ Si el parámetro status es una cadena, la muestra antes de salir.
  - ✓ Si es un número no se muestra.
  - ✓ El 0 se usa para indicar la finalización del programa de forma satisfactoria.

- La función **die()** es equivalente a exit(), finaliza la ejecución del script
- La función **header(),** muy utilizada para redirecciones en php a través de Location

```
if (!isset($numero)){
        die();
```

DIFERENCIAS:
exit () simplemente rescata el programa con un estado de salida numérico, mientras que **die** () imprime el mensaje de error en stderr y sale con el estado EXIT_FAILURE.