Conceptos Básicos php II. Arrays

Heike Bonilla Redondo

Índice

1. Arrays

Hasta ahora hemos trabajado en nuestros programas con una pequeña cantidad de datos, y sin que la cantidad de los mismos variará a lo largo del programa. Sin embargo, la mayoría de los programas útiles requieren tratar gran cantidad de datos. En este caso sería muy incómodo tener una variable para cada dato.

Afortunadamente, cuando se tratan muchos datos, éstos suelen ser de unos tipos muy similares entre sí. Para tratarlos en grupo, *PHP* ofrece tipos de datos compuestos. Es decir, tipos de datos que permiten almacenar varios datos en una misma variable.

El tipo de datos compuesto más sencillo es el *array*. Es una estructura muy potente, flexible y de uso muy intuitivo. Un array está compuesto por varios *elementos*. Cada *elemento* almacena un *valor* diferente. Estos elementos se almacenan como pares clave-valor. De hecho, puedes usar un arreglo cada vez que haya la necesidad de almacenar una lista de elementos. La mayoría de las veces, todos los elementos de un arreglo tienen tipos de datos similares.

2. Arrays indexados numéricamente.

Estos elementos se almacenan como pares clave-valor. De hecho, puedes usar un arreglo cada vez que haya la necesidad de almacenar una lista de elementos. La mayoría de las veces, todos los elementos de un arreglo tienen tipos de datos similares.

El tipo más sencillo de arrays son los indexados numéricamente, en los que el índice de cada elemento corresponde con su posición. El primer elemento de un array tiene índice 0, y no 1 como cabría esperar. De igual modo, el último elemento corresponde al indice longitud-1.

2.1. INICIALIZAR UN ARRAY

Los arrays se suelen almacenar en variables, como cualquier otro tipo de datos.

Dos maneras de **crear un array vacio**, inicializarlo:

```
<?php
   $array = array();
?>
```

```
<?php
   $array = array[];
?>
```

Tenemos también varias maneras de insertar elementos a un array:

```
Ejemplo Creación de un array por asignación directa mediante la función array():
<?php
$edades = array(28, 43, 32, 55);
$formas = array("triángulo", "cuadrado", "círculo");
?>
```

La segunda forma de rellenar un array es añadiéndole elementos al final del array. Para añadir un elemento a un array se usa su identificador seguido de corchetes "[ ]" sin índice, y se le asigna un valor.

```
Ejemplo Aquí se puede ver como se crea un array mediante la adición de elementos.
<?php
$paises[] = "Italia"; //Añade el elemento con índice 0
```

```
$paises[] = "Francia"; //Añade el elemento con índice 1
$paises[] = "Portugal"; //Añade el elemento con índice 2
?>
```

Ejemplo. Se pueden combinar ambas formas, primero declarar un array directamente y cuando sea necesario ir añadiendo elementos.

```
<?php
$frutas = array("melón", "sandía", "naranja");
$frutas[] = "manzana";
$frutas[] = "melocoton";
?>
```

2.2. ACCEDER A UN ELEMENTO DE UN ARRAY.

Necesitaremos acceder a los elementos de un array con el fin de asignarles valores o de leer su contenido. Esto es muy sencillo, basta poner el índice del elemento al que queremos acceder entre corchetes "[...]".

Ejemplo. Aquí se leen los valores de los elementos de un array para sacarlos por pantalla. Luego se asigna un nuevo valor al primer elemento (recordamos, con índice 0).

```
<?php
$frutas = array("melón", "sandía", "naranja");
echo "La primera fruta de la lista es $frutas[0].<br>"; echo "La segunda fruta de la lista es $frutas[1].<br>"; echo "La tercera fruta de la lista es $frutas[2].<br>";
$frutas[0] = "limón"; //Asigna un nuevo valor al elemento 1
echo "La primera fruta de la lista es $frutas[0].";
?>
```

Si se solicita un valor no definido de una matriz, se produce un aviso (undefined offset). Recuerda, los avisos no interrumpen la ejecución del programa, pero se deben corregir porque el programa seguramente no tendrá el comportamiento esperado.

2.3. RECORRER UN ARRAY.

Los arrays se suelen utilizar para almacenar listas de valores. Por ello, una de las acciones más habituales que se hacen con ellos es recorrerlos de principio a fin para leer o modificar uno o varios de ellos.

Los tres métodos siguientes nos permiten recorrer arrays para poder manipular las claves y los valores de un array:

- **Utilizando la estructura de control for**

Ejemplo Recorriendo un array para mostrarlo por pantalla.

```
<?php
$frutas = array("melón", "sandía", "naranja");
for ($i = 0; $i < count($frutas); $i++) {
echo "Elemento $i: $frutas[$i]<br>";
}
?>
```

Antes de utilizar este método para hacer el recorrido hay que tener en cuenta que:

  - Se limita a arrays numéricos.
  - Los *keys* pueden estar desordenados.
  - Puede faltar algún *key*, por lo que genera un error del tipo *Undefined offset*.

- **Utilizando la estructura de control foreach**

```
<?php
   foreach ($array as $val) {
            echo $val;
   }
?>
```

Este método es mucho más rápido y fácil de utilizar. En este ejemplo $val es el **valor** de cada *key* que pasa por la iteración, por lo que las *keys* quedan ignoradas.

Para extraer también las *keys*:

```
<?php
            foreach($array as $key => $val) {
            print "$key = $val <br>";
            }
?>
```

En este caso también podemos utilizar las *keys*, especialmente útil cuando son **arrays asociativos** que veremos más adelante.

3. Arrays no secuenciales

Hasta ahora lo arrays que hemos visto tenían como índices la sucesión 0 (1er elemento), 1 (2o elemento), 2 (3er elemento), y así sucesivamente. En *PHP* los índices de un array no

tienen por qué ser consecutivos, pueden incluso estar desordenados.

El índice de los arrays asociativos no tiene por qué ser necesariamente un entero, puede ser también un número decimal o una cadena.

4. ARRAYS ASOCIATIVOS

Éstos son un caso específico de arrays no secuenciales, en los que el índice es una cadena de texto. Pueden resultar útiles para guardar listas, en las que se asocia un valor a una palabra clave.

4.1. INICIALIZAR ARRAY ASOCIATIVO

La creación de este tipo de arrays se puede hacer de dos formas. La primera es mediante la función *array()* de forma parecida a como lo hacíamos antes, solo que ahora deberemos especificar un valor para el índice.

Ejemplo Cuando se declaran explícitamente los índices se escribe el valor del índice seguido por "=>" y el valor del elemento que contiene.

```
<?php
$capitales = array("Italia" => "Roma",
"Francia" => "Paris", "Portugal" => "Lisboa");
?>
```

La segunda forma de crear un array asociativo es añadiendo elementos al array, y asignándoles de forma explícita cual es su índice.

Ejemplo Veamos esta segunda forma de inicializar arrays declarando explícitamente sus índices.

```
<?php
$alturas["Aneto"] = 3404;
$alturas["Teide"] = 3718;
$alturas["Mulhacen"] = 3748;
?>
```

4.2. ACCESO ELEMENTOS ARRAY ASOCIATIVO

- **Acceso puntual a elementos del array**

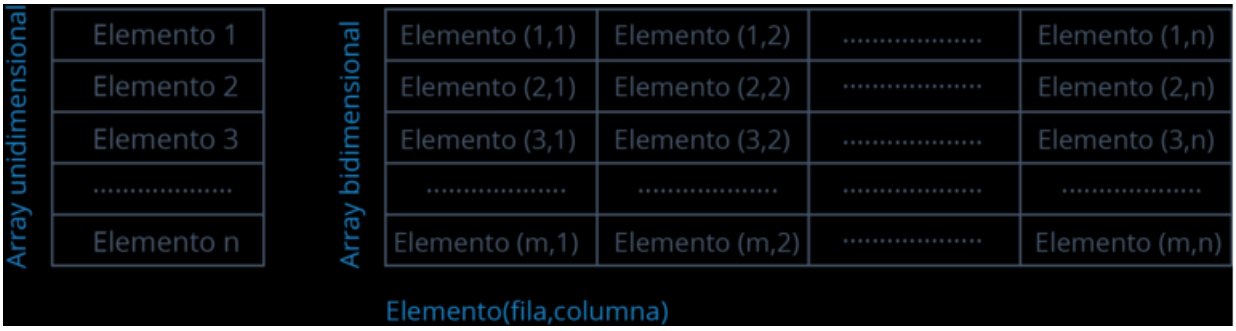
```
<?php
$capitales = array(Francia=>'Paris', Alemania=>'Berlin', Italia=>'Roma');
echo $capitales[“Francia”]; // nos devolverá París
?>
```

- **Recorrido de array asociativo con foreach**

```
<?php
$capitales = array(Francia=>'Paris', Alemania=>'Berlin', Italia=>'Roma');
foreach($capitales as $pais => $capital){
   echo "La capital de $pais es: $capital<br>";
}
?>
```

5. Arrays multidimensionales

Un **array multidimensional** es un array en cuyo interior tiene otro array. Este, a su vez, puede tener otro y así sucesivamente hasta un número de dimensiones determinado. Consideramos la **dimensión** como el nivel de anidamiento o de inclusión de arrays que se alcanza, de manera que un **array multidimensional de dimensión 2 o array bidimensional** está compuesto de un array que dentro tiene otros arrays y estos últimos no tienen más arrays dentro.



5.1. Multidimensional de índice numérico

Al igual que ocurre en el unidimensional, cada elemento ocupa una posición en el array, pero al ser bidimensional requeriremos de dos posiciones para acceder al mismo, la primera hará referencia a la fila y la segunda a la columna, veamos un ejemplo de su sintaxis:

- **Inicializar array multidimensional**

```
<?php
$usuarios = array(
array("Nacho", "valencia"),
array("Jaime", "Madrid"),
array("Maria", "Barcelona")
);
?>
```

- **Acceso a un array bidimensional indexado en php:**

```
<?php
$usuarios = array(
array("Nacho", "valencia"),
array("Jaime", "Madrid"),
array("Maria", "Barcelona")
);
echo $usuarios[2][1]; // accederíamos a María,
echo $usuarios[0][1]; // accederíamos a Valencia
?>
```

- **Recorrido de un array bidimensional indexado en php:**

Igual que ocurre con los arrays unidimensionales podemos hacerlo con bucles for o con foreach, siempre más recomendable el uso de foreach. En este caso tendremos que anidar bucles para realizar el recorrido completo.

```
<?php
$usuarios = array(
array("Nacho", "valencia"),array("Jaime", "Madrid"),
array("Maria", Barcelona")
);
//Recorrido con foreach

foreach($usuarios as $pareja=>$valor)
{
   echo "La pareja número $pareja: <br>";
   foreach($valor as $usuario)
   {
      echo $usuario ." <br>";
   }
   echo "<br>";
}
echo "<br>";
?>
```

Ejemplo Se muestra como almacenar una matriz, o array de dos dimensiones, que contiene una sopa de letras generada aleatoriamente, mediante la función chr(), que devuelve un carácter dado su número ascii.

```
<?php DEFINE("ALTO",10); DEFINE("ANCHO",20);
//recorrido para llenar matriz
$sopa_letras = array();
for ($i = 0; $i < ALTO; $i++) {
for ($j = 0; $j <ANCHO; $j++) {
$sopa_letras[$i][$j] = chr(rand(65, 90));
}
}
echo "SOPA DE LETRAS<br>";
echo "<br>";
//recorrido para mostrar
for ($i = 0; $i < ALTO; $i++) {
for ($j = 0; $j <ANCHO; $j++) {
echo $sopa_letras[$i][$j];
}
echo "<br>";
}
?>
```

5.2. Array Bidimensional Asociativo

- **Insertar y mostrar información en array asociativo**

```
<!--Ejemplo de array bidimensional asociativo .-->

<?php
$jugadores = array (
   array(
      'nombre' => 'Messi, Lionel',
      'equipo' => 'Barcelona'
   ),
   array(
      'nombre' => 'Ronaldo, Cristiano',
      'equipo' => 'Real Madrid'
   ),
   array(
      'nombre' => 'Saturno, Sergio',
      'equipo' => 'Boca Juniors'
   ),
   array(
      'nombre' => 'Neymar',
      'equipo' => 'Santos'
   ),
```

```
);
// accederemos a Messi, Lionel
echo $jugadores[0]['nombre'];
?>
```

- **Recorrido array bidimensional asociativo**

No podemos usar *for* con índices asociativos, puesto a que las claves de los elementos son cadenas de caracteres o números desordenados y no números enteros ordenados.

```
<?php
 $array = ['Luis'=>['piso'=>5, 'escalera'=>'A', 'puerta'=>15],
        'Ana'=>['piso'=>2, 'escalera'=>'A', 'puerta'=>5],
        'Juan'=>['piso'=>3, 'escalera'=>'B', 'puerta'=>10]];

// Mostramos contenido del array multidimensional con la clave asociada a  cada dimensión y los valores de sus elementos
 foreach($array as $inquilino=>$datos):
   echo "<p>Inquilino:  $inquilino<br>";
   foreach($datos as  $dato=>$valor) {
     echo "$dato: $valor<br>";
   }
   echo "</p>";
 endforeach;
?>
```

6. Algunas funciones para manejo de Arrays

Algunas funciones útiles: count, print_r, unset, y array_values

- La función count sirve para “contar” la cantidad de elementos que tiene un array.
- La función print_r sirve para mostrar un array. Rara vez se utiliza en versiones finales, pero suele ser muy útil para depuración, pues es una forma rápida de ver el contenido del array.
- La función unset “desasigna” un elemento de un array. Los arrays no se reindexan, simplemente quedan sin un elemento.
- La función array_values, “reordena” las claves del array (0 para el primero, 1 para el segundo, etc).

```
//Defino el array $a, con 5 elementos
$a=array(2,4,6,8,10);
$cant=count($a);
//$cant vale 5, puesto que esa es la cantidad de elementos que tiene $a.
print_r($a);
//Muestra:
// array([0]=>2, [1]=>4, [2]=>6, [3]=>8, [4]=>10)
unset($a[3]); //Elimina el cuarto elemento de $a
print_r($a);
//Muestra:
// array([0]=>2, [1]=>4, [2]=>6, [4]=>10)
$b=array_values($a);
//$a no cambia, pero se asigna en $b el array $a "reordenado":
print_r($b);
//Muestra:
// array([0]=>2, [1]=>4, [2]=>6, [3]=>10)
```