Entorns de Desenvolupament

9. Arrays

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

IES DR. LLUÍS SIMARRO

9. Arrays

Un **array** és un **conjunt de valors independents que es guarden en una mateixa variable**. En PHP ni tan sols és necessari que siguen totes del mateix tipus, tot i que el més habitual és que sí que ho siguen. S'**accedeix** a cada **element individual** de l'array mitjançant un nombre enter anomenat **índex**. **0** és l'índex o localitzador del **primer element** i **n-1** és l'índex de l'**últim element**, essent **n** la **dimensió**, **grandària**, **mida** o **longitud** de l'array. Així `$empleat[23]` representaria l'empleat amb localitzador 23 com veurem a continuació.

Els arrays són molt utilitzats en programació. Depenent de la quantitat de dimensions que tinguen poden ser:

- D'una dimensió (**vectors**). Per exemple `$empleat[23]`
- De dues dimensions (**matrius**). Per exemple `$butaca[3][14]`
- De tres o més dimensions (**multidimensionals**). Per exemple `$data[2021][12][1]`

Per **accedir** als **elements** de l'array s'utilitzen els **claudàtors [ ]**, dins dels quals hi haurà un **localitzador** o **índex** que és un **nombre sencer**. A més, podem **guardar valors** de **qualsevol tipus** de variable (string, enter, decimal (float), booleà) dins d'un array. En el mateix array, poden haver guardats valors de diferents tipus simultàniament. Podem tindre un booleà en una posició i un string en una altra del mateix array:

```
$valors[0] = "Xàtiva"; $valors[1] = true; $valors[2] = 7.5;
```

Per exemple `$empleat[23]` podria prendre com a valor `$empleat[23] = "Joan Martí Suárez"`. En aquest cas es tractaria d'un **array de cadenes de text**, és a dir, una matriu de string. Altrament `$empleat[23]` podria prendre com a valor `$empleat[23] = 2312`. En aquest cas es tractaria d'un **array de valors numèrics**.

En alguns llenguatges cal **declarar** els **arrays abans** de poder **utilitzar-los**, però en **PHP no cal**. Quan es defineixen elements d'un array, **PHP reconeix automàticament** que es tracta d'un **array** sense necessitat de declaració prèvia.

Vectors (Arrays d'una dimensió)

Els vectors són els arrays que només contenen una dimensió (un índex). Veiem el següent **exemple 01**:

```
<?php
      $estacio[0] = "Primavera";
      $estacio[1] = "Estiu";
      $estacio[2] = "Tardor";
      $estacio[3] = "Hivern";
      echo $estacio[2];
      $nom[0] = 7;
      $nom[1] = 11;
      $nom[2] = 15;
      echo $nom[1];
?>
```

La forma general d'ús d'un array és:

```
$nomDeVariableArray [índex] = valorAsignat;
```

S'ha de tindre en compte que el que diferencia una variable que pertany a un array (conjunt de variables) és la **presència del claudàtor** amb un índex al seu interior.

```
$Jugador[8] ----> és una variable amb índex.
$Jugador8 ----> és una variable normal.
$TCP[3] ----> és una variable amb índex.
$TCP3 ----> és una variable normal.
```

PHP admet també una altra **possibilitat** que no admeten altres llenguatges: es poden **usar arrays sense especificar els números dels índexs**, ja que PHP els pot posar automàticament.

Exemple 02:

```
<?php
	$ciutat[] = "Xàtiva" ;
	$ciutat[] = "Canals" ;
	$ciutat[] = "Genovés" ;
	$ciutat[] = "La Llosa" ;
	echo $ciutat[3];
?>
```

En aquest exemple **PHP** ha **col·locat automàticament** els **índexs**, assignant el 0 a Xàtiva, l'1 a Canals, el 2 a Genovés i el 3 a La Llosa. Cal recordar que els arrays sempre comencen numerar des de zero.

Una altra forma d'assignar valors a un array vector és la següent:

```
<?php
	$color = array("blau" , "verd" , "negre" , "marró");
	echo $color[1];
?>
```

La forma general és:

```
$nomDelArray=array(valorElement0, valorElement1,..., valorElementn);
```

El terme **array** és una **paraula clau** en **PHP** el significat és equivalent a dir "**els elements que apareixen a la llista a continuació són elements d'un array**".

Es pot també assignar un conjunt de valors a un array en una sola declaració amb els claudàtors:

```
<?php
      $color = ["roig" , "blau" , "groc" , "verd"];
      echo $color[1];
?>
```

Inclús hi ha encara una altra manera de declarar diversos elements d'un array, utilitzant l'operador `=>`, indicant l'índex i el valor:

```
<?php
   $color = array(0 => "roig" , 1 => "blau" ,2 => "groc" ,3 => "verd"];
   echo $color[1];
?>
```

Una funció interessant per poder **visualitzar** tot el contingut d'un array és `var_dump`, que ens evita haver de fer un bucle per recórrer-ne tots els elements.

Exemple:

```
<?php
      $alumnes=array("Salva" , "Rebeca" , "Anna" , "Maria");
      //visualitzem el contingut de la variable $alumnes
      var_dump($alumnes);
?>
```

Es visualitzaria:

```
array(4) { [0]=> string(5) "Salva" [1]=> string(6) "Rebeca" [2]=> string(4) "Anna"
[3]=> string(5) "Maria" }
```

Eliminar elements d'un array

Mètode unset()

Quan s'utilitza `unset()` els índex de l'array no canviaran ni es reindexaran. Si volem reindexar els elements podem usar `array_values()` després d'aplicar `unset()`,

fet que convertirà tots els índex a valors numèrics numerats a partir de 0. Si estem utilitzant arrays associatius, canviarà també els noms a valors enters consecutius.

Exemple:

```
<?php
    $array = array("a","b","c");
     //altra manera de crear-lo: $array = [0 => "a", 1 => "b", 2 => "c"];
    unset($array[1]); //Sent 1 l'índex de l'element que volem eliminar
    var_dump($array);
    //reindexem i visualitzem
    $array=array_values($array);
    var_dump($array);
?>
```

Com a resultat, la variable `$array` contindria, abans d'aplicar `array_values()` (havent eliminat l'element en la posició 1):

```
[
  [0] => a
  [2] => c
]
```

Després d'aplicar `array_values()` (reindexant els elements des de 0):

```
[
  [0] => a
  [1] => c
]
```

Mètode array_splice()

Si utilitzem `array_splice()`, els índex es tornaran a indexar automàticament, però si treballem amb arrays associatius, les claus associatives no canviaran, a diferència del cas en què utilitzem `array_values()`, en què convertiria tots els índex a valors numèrics.

`array_splice()` pot tindre fins a 4 paràmetres:

array_splice(array_entrada, offset, longitud_a_eliminar, array_reemplaçaments);

- **array_entrada**: array sobre el que es vol treballar.
- **offset**: No és l'índex de l'element, és el **desplaçament respecte a l'inici de l'array de l'element que volem eliminar**. Si és positiu, comença des de l'inici, si és negatiu, comença des del final de l'array.
- **longitud_a_eliminar** (opcional): si no s'indica, s'elimina tot des de la posició de l'offset fins al final. Si s'indica i és positiu, s'elimina aquesta quantitat d'elements a partir de la posició de l'offset. Si s'especifica i és negatiu, aleshores, el final dels elements eliminats serà a tantes posicions del final de l'array com indique el número.
- **array_reemplaçaments** (opcional): Si s'especifica, els elements eliminats seran substituïts per aquests elements.

Exemple:

```
<? php
      $array = array("a","b","c");
      //eliminem un element a partir de la  posició 1 de l'array:
      array_splice ($array, 1, 1);
      var_dump($array);
?>
```

En aquest cas, els índex es desplacen, substituint aquell que s'ha eliminat pel que el segueix i així successivament. Com a resultat, la variable `$array` contindria:

```
[
   [0] => a
   [1] => c
]
```

Tant `array_splice()` com `unset()` prenen el valor de l'array **per referència**, això vol dir que no cal assignar els valors de retorn d'aquestes funcions de nou a l'array, es modifica directament.

Si volem esborrar diversos elements de l'array i no volem cridar `unset()` o `array_splice()` diverses vegades, podem utilitzar les funcions `array_diff()` o `array_diff_key()` depenent de si coneixem els valors o les claus dels elements que volem suprimir.

Mètode array_diff()

Si coneixem els valors dels elements de l'array que volem suprimir, podem utilitzar `array_diff()`. Com abans amb `unset()` no canviarà ni reindexarà els índex de l'array. En aquest cas sí que cal assignar el resultat a la variable `$array`, perquè s'agafa el valor `$array` **per valor**, no per referència.

Exemple:

```
<? php
      $array = array("a","b","c");
      $array = array_diff($array, ["a", "c"]); // indiquem els valors
                                                que volem suprimir
      var_dump($array);
?>
```

Com a resultat, la variable `$array` contindria:

```
[
  [1] => b
]
```

Mètode array_diff_key()

Si coneixem els índex dels elements que volem suprimir, podem utilitzar `array_diff_key()`. Ens hem d'assegurar de passar aquestes claus a suprimir com a claus en el segon array que es passa com paràmetre i no com a valors. Els índex no es canviaran ni reindexaran. En realitat elimina del primer array tots els índex que no coincideixen amb el segon array.

Exemple:

```
<? php
    $array = array("a","b","c");
    $array = array_diff_key($array, [0 => "xy", "2" => "xy"]);
     // volem  eliminar els elements amb índex 0 i 2, els valors "xy"
     són totalment irrellevants.
   var_dump($array);
?>
```

Eixida de l'operació, igual que en el cas anterior.

```
[
  [1] => b
]
```

Ací teniu un llistat complet dels mètodes incorporats en php per treballar amb arrays:

https://www.php.net/manual/en/ref.array.php