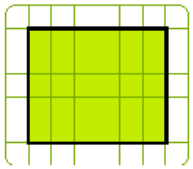
Ordenar matrices multidimensionales en PHP



En un tutorial de PHP vimos cómo ordenar matrices. En este artículo vamos a conocer un recurso que PHP nos ofrece cuando se trata de ordenar matrices multidimensionales, manteniendo, por supuesto, la integridad de los datos originales.

ORDENAR MATRICES MULTIDIMENSIONALES

Este es un tema peliagudo en muchos lenguajes de programación. Se trata de ordenar los elementos de una matriz bidimensional (que, cómo ya sabes, son, a su vez, matrices unidimensionales), por un criterio. Por ejemplo, en las matrices que hemos visto de amigos, podemos querer ordenarlas por el nombre. Afortunadamente, PHP cuenta con la función array_multisort(), concebida para hacer, precisamente, eso. Vamos a ver cómo, en un ejemplo que hemos llamado ordenarMulti.php (la imaginación al poder :-D).

```
<?php
 $matrizDeAmigos = array (
  array("nombre"=>"Pedro Torres", "direccion"=>"CL Mayor, 37", "telefono"=>123456789),
  array("nombre"=>"Carlos Gómez", "direccion"=>"CL Alfareros, 12", "telefono"=>567891234),
  array("nombre"=>"Susana Casas", "direccion"=>"CL Sierra Grande, 2", "telefono"=>987654321),
  array("nombre"=>"Carmen Pérez", "direccion"=>"CL Himalaya, 189", "telefono"=>502983948)
 );

 $nombre = array();
 foreach ($matrizDeAmigos as $keyAmigo=>$amigo) {
  $nombre[$keyAmigo] = $amigo["nombre"];
 }
 array_multisort($nombre, SORT_ASC, $matrizDeAmigos);
 echo "<pre>";
 var_dump($matrizDeAmigos);
?>
```

Lo que queremos hacer es ordenar la matriz de amigos por el nombre, de modo que, por supuesto, se mantenga la relación del nombre con el resto de los datos, es decir, que se mantenga lo que se conoce cómo **la integridad de datos**. Para ello creamos una matriz auxiliar que, en nuestro ejemplo, hemos llamado $nombre. El proceso que aparece con la instrucción foreach probablemente no lo entiendas totalmente ahora, hasta que no publiquemos el artículo sobre estructuras de control de flujo pero, básicamente, lo que hace es recorrer la matriz de amigos y rellenar la matriz auxiliar $nombre con los nombres de los amigos, asignando a cada uno la clave que el amigo tiene en la matriz original. Para que te hagas idea de lo que estamos haciendo, al terminar el bucle foreach la matriz auxiliar $nombre tendrá el siguiente contenido:

CLAVE
VALOR

0
Pedro Torres

1
Carlos Gómez

2
Susana Casas

3
Carmen Pérez

La forma en que se logra esto, como te he comentado, la veremos en detalle cuando hablemos de estructuras de control de flujo.
A lo que vamos. La función array_multisort() toma tres argumentos. El primero es la matriz auxiliar que, como hemos visto, se construye a partir del dato por el que queremos ordenar. El segundo es una constante propia de PHP, para indicar si la ordenación se va a realizar de forma ascendente o desdendente: puede ser SORT_ASC o SORT_DESC. El tercer argumento es la matriz bidimensional que vamos a ordenar siguiendo el criterio elegido. Al ejecutarse esta función la matriz bidimensional queda ordenada, tal como nos muestra en pantalla el script del ejemplo.
Pero array_multisort() es más potente que esto. Nos permite ordenar matrices por más de un índice, simultáneamente. Aclaremos esto. Supón que tienes una matriz, digamos, de empleados de una empresa. Cada uno de estos empleados tiene un rango, un nombre, unos apellidos y otros dstos adicionales. El rango puede ser uno de tres posibles: directivo, mando intermedio y empleado. La matriz podría tener una estructura similar a la siguiente:

```
$empleados = array(
 array(
  "rango"=>"Rango del primer empleado",
  "apellidos"=>"Apellidos del primer empleado",
  "nombre"=>"Nombre del primer empleado",
  "departamento"=>"Departamento del primer empleado",
  "salario"=>"Salario del primer empleado",
  "fecha_ingreso"=>"Fecha de ingreso del primer empleado"
 ),
 array(
  "rango"=>"Rango del segundo empleado",
  "apellidos"=>"Apellidos del segundo empleado",
  "nombre"=>"Nombre del segundo empleado",
  "departamento"=>"Departamento del segundo empleado",
  "salario"=>"Salario del segundo empleado",
  "fecha_ingreso"=>"Fecha de ingreso del segundo empleado"
 ),
 array(
  "rango"=>"Rango del tercer empleado",
  "apellidos"=>"Apellidos del tercer empleado",
  "nombre"=>"Nombre del tercer empleado",
  "departamento"=>"Departamento del tercer empleado",
  "salario"=>"Salario del tercer empleado",
  "fecha_ingreso"=>"Fecha de ingreso del tercer empleado"
 ),

 ...
```

```
 array(
  "rango"=>"Rango del empleado N",
  "apellidos"=>"Apellidos del empleado N",
  "nombre"=>"Nombre del empleado N",
  "departamento"=>"Departamento del empleado N",
  "salario"=>"Salario del empleado N",
  "fecha_ingreso"=>"Fecha de ingreso del empleado N"
 ),
);
```

Lo que queremos es que se nos ordena la matriz por rango pero, dentro de cada rango, por apellidos y nombre (en esa secuencia). Nada más fácil. Observa las siguientes líneas de código:

```
$rangos = array();
$apellidos = array();
$nombres = array();
foreach ($empleados as $clave=>$empleado){
 $rangos[$clave] = $empleado["rango"];
 $apellidos[$clave] = $empleado["apellidos"];
 $nombres[$clave] = $empleado["nombre"];
}
array_multisort($rangos, $apellidos, $nombres, SORT_ASC, $empleados);
```

Como ves. antes de la constante que establece el sentido de la ordenación, podemos indicar todos los campos necesarios, siempre que hayamos creado las correspondientes matrices en el bucle previo.

Otro ejemplo, en el que vemos que no sólo se puede ordenar por varios criterios, si no, además, hacer que algunos sean ascedentes y otros descendentes.

```
<?php
echo "<pre>";
$people = [
    ['name' => 'Antonio', 'age' => 22, 'points' => 600],
    ['name' => 'Luis', 'age' => 21, 'points' => 439],
    ['name' => 'Luis', 'age' => 23, 'points' => 435],
    ['name' => 'Luis', 'age' => 23, 'points' => 200],
    ['name' => 'Marta', 'age' => 19, 'points' => 175],
    ['name' => 'Sonia', 'age' => 43, 'points' => 800],
    ['name' => 'Sonia', 'age' => 43, 'points' => 700],
    ['name' => 'Sonia', 'age' => 25, 'points' => 356],
    ['name' => 'Marta', 'age' => 19, 'points' => 788],
    ['name' => 'Alfredo', 'age' => 19, 'points' => 239],
];
function MultiSorting($matriz)
{
 foreach ($matriz as $clave => $person) {
  $name[$clave] = $person['name'];
  $age[$clave] = $person['age'];
  $points[$clave] = $person['points'];
 }

 array_multisort($name, SORT_ASC,
   $age, SORT_DESC,
   $points, SORT_ASC,
```

```
  $matriz
 );
 return $matriz;
}
var_dump(MultiSorting($people));
?>
```