Focus search box

array_unshift »
« array_uintersect

- Manual de PHP
- Referencia de funciones
- Extensiones relacionadas con variable y tipo
- Arrays
- Funciones de Arrays

Change language: Spanish

Submit a Pull Request Report a Bug

array_unique

(PHP 4 >= 4.0.1, PHP 5, PHP 7, PHP 8)

array_unique — Elimina valores duplicados de un array

Descripción ¶

array_unique(array `$array`, int `$sort_flags` = SORT_STRING): array

Toma un `array` y devuelve un nuevo array sin valores duplicados.

Tenga en cuenta que las claves se conservan. Si múltiples elementos se comparan bajo el parámetro `sort_flags`, entonces la clave y el valor del primer elemento igual se conservarán.

> **Nota**: Dos elementos son considerados iguales solo si `(string) $elem1 === (string) $elem2`, es decir, cuando la representación en formato de string sea la misma, se usará el primer elemento.

Parámetros ¶

`array`

El array de entrada.

`sort_flags`

El segundo parámetro opcional `sort_flags` se puede utilizar para modificar el tipo de orden usando estos valores:

Indicadores de tipos de orden:

- **`SORT_REGULAR`** - compara ítems normalmente (no cambia los tipos)
- **`SORT_NUMERIC`** - compara ítems numéricamente
- **`SORT_STRING`** - compara ítems como strings
- **`SORT_LOCALE_STRING`** - compara ítems como strings, basados en la configuración regional en uso.

Valores devueltos ¶

Devuelve el array filtrado.

Historial de cambios ¶

Versión	Descripción
7.2.0	Si el parámetro `sort_flags` es **`SORT_STRING`**, el `array` anterior ha sido copiado y se han eliminado los elementos no únicos (sin empaquetar el array posteriormente), pero ahora se construye un nuevo array añadiendo los elementos únicos. Esto puede dar lugar a diferentes índices numéricos.
5.2.10	Se volvió a cambiar el valor predeterminado de `sort_flags` a **`SORT_STRING`**.
5.2.9	Se añadió el parámetro opcional `sort_flags` con el valor predeterminado **`SORT_REGULAR`**. Antes de 5.2.9, esta función se usaba para ordenar el array con **`SORT_STRING`** internamente.

Ejemplos ¶

Ejemplo #1 Ejemplo de array_unique()

```php
<?php
$entrada = array("a" => "verde", "rojo", "b" => "verde", "azul", "rojo");
$resultado = array_unique($entrada);
print_r($resultado);
?>
```

El resultado del ejemplo sería:

```
Array
(
    [a] => verde
    [0] => rojo
    [1] => azul
)
```

Ejemplo #2 array_unique() y tipos

```php
<?php
$entrada = array(4, "4", "3", 4, 3, "3");
$resultado = array_unique($entrada);
var_dump($resultado);
?>
```

El resultado del ejemplo sería:

```
array(2) {
  [0] => int(4)
  [2] => string(1) "3"
}
```

Ver también ¶

- array_count_values() - Cuenta todos los valores de un array

Notas ¶

> **Nota**: Observe que **array_unique()** no está pensado para que trabaje con arrays multidimensionales.

+ add a note

User Contributed Notes 29 notes

up
down
289

Ghanshyam Katriya(anshkatriya at gmail) ¶

7 years ago

Create multidimensional array unique for any single key index.
e.g I want to create multi dimentional unique array for specific code

Code :
My array is like this,

```
<?php
$details = array(
    0 => array("id"=>"1", "name"=>"Mike",    "num"=>"9876543210"),
    1 => array("id"=>"2", "name"=>"Carissa", "num"=>"08548596258"),
    2 => array("id"=>"1", "name"=>"Mathew",  "num"=>"784581254"),
);
?>
```

You can make it unique for any field like id, name or num.

I have develop this function for same :

```
<?php
function unique_multidim_array($array, $key) {
    $temp_array = array();
    $i = 0;
    $key_array = array();

    foreach($array as $val) {
        if (!in_array($val[$key], $key_array)) {
            $key_array[$i] = $val[$key];
            $temp_array[$i] = $val;
        }
        $i++;
    }
    return $temp_array;
}
?>
```

Now, call this function anywhere from your code,

something like this,

```
<?php
$details = unique_multidim_array($details,'id');
?>
```

Output will be like this :

```
<?php
$details = array(
    0 => array("id"=>"1","name"=>"Mike","num"=>"9876543210"),
    1 => array("id"=>"2","name"=>"Carissa","num"=>"08548596258"),
);
?>
```

up
down
16

falundir at gmail dot com ¶

4 years ago

I find it odd that there is no version of this function which allows you to use a comparator callable in order to determine items equality (like array_udiff and array_uintersect). So, here's my version for you:

```
<?php
function array_uunique(array $array, callable $comparator): array {
    $unique_array = [];
    do {
        $element = array_shift($array);
        $unique_array[] = $element;

        $array = array_udiff(
            $array,
            [$element],
            $comparator
        );
    } while (count($array) > 0);

    return $unique_array;
}
?>
```

And here is a test code:

```
<?php
class Foo {

    public $a;

    public function __construct(int $a) {
        $this->a = $a;
    }
}

$array_of_objects = [new Foo(2), new Foo(1), new Foo(3), new Foo(2), new Foo(2), new Foo(1)];

$comparator = function (Foo $foo1, Foo $foo2): int {
    return $foo1->a <=> $foo2->a;
};

var_dump(array_uunique($array_of_objects, $comparator)); // should output [Foo(2), Foo(1), Foo(3)]
?>
```

up
down
17
stoff@_¶
5 years ago

In reply to performance tests array_unique vs foreach.

In PHP7 there were significant changes to Packed and Immutable arrays resulting in the performance difference to drop considerably. Here is the same test on php7.1 here;
http://sandbox.onlinephpfunctions.com/code/2a9e986690ef8505490489581c1c0e70f20d26d1

```
$max = 770000; //large enough number within memory allocation
$arr = range(1,$max,3);
$arr2 = range(1,$max,2);
$arr = array_merge($arr,$arr2);

$time = -microtime(true);
$res1 = array_unique($arr);
$time += microtime(true);
```

```
echo "deduped to ".count($res1)." in ".$time;
// deduped to 513333 in 1.0876770019531

$time = -microtime(true);
$res2 = array();
foreach($arr as $key=>$val) {
    $res2[$val] = true;
}
$res2 = array_keys($res2);
$time += microtime(true);
echo "<br />deduped to ".count($res2)." in ".$time;
// deduped to 513333 in 0.054931879043579
```

up
down
92

Anonymous ¶

12 years ago

It's often faster to use a foreache and array_keys than array_unique:

```
    <?php

    $max = 1000000;
    $arr = range(1,$max,3);
    $arr2 = range(1,$max,2);
    $arr = array_merge($arr,$arr2);

    $time = -microtime(true);
    $res1 = array_unique($arr);
    $time += microtime(true);
    echo "deduped to ".count($res1)." in ".$time;
    // deduped to 666667 in 32.300781965256

    $time = -microtime(true);
    $res2 = array();
    foreach($arr as $key=>$val) {
        $res2[$val] = true;
    }
    $res2 = array_keys($res2);
    $time += microtime(true);
    echo "<br />deduped to ".count($res2)." in ".$time;
    // deduped to 666667 in 0.84372591972351

    ?>
```

up
down
6

Fabiano ¶

4 years ago

As for PHP 7.1.12, this is the comparison between array_keys(array_flip()), array_flip(array_flip()), for each elimination and array_unique. The array_keys(array_flip()) is the fastest method to remove duplication values from a single dimension array:

```
<?php

$max = 1000000;
$arr = range(1,$max,3);
$arr2 = range(1,$max,2);
$arr = array_merge($arr,$arr2);
```

```
$time = -microtime(true);
$res1 = array_unique($arr);
$time += microtime(true);

echo "<br>deduped to ".count($res1)." in ".$time;
// deduped to 666667 in 0.78185796737671
// memory used: 33558528

$time = -microtime(true);
$res2 = array_flip(array_flip($arr));
$time += microtime(true);

echo "<br><br>deduped to ".count($res2)." in ".$time;
// deduped to 666667 in 0.072191953659058
// memory used: 3774873

$time = -microtime(true);
$res3 = array();
foreach($arr as $key=>$val) {
    $res3[$val] = true;
}
$res3 = array_keys($res3);
$time += microtime(true);

echo "<br /><br>deduped to ".count($res3)." in ".$time;
// deduped to 666667 in 0.095494985580444
// memory used: 33558528

$time = -microtime(true);
$res4 = array_keys(array_flip($arr));
$time += microtime(true);

echo "<br /><br>deduped to ".count($res4)." in ".$time;
// deduped to 666667 in 0.05807900428772
// memory used: 33558528
```

up
down
22

Ray dot Paseur at SometimesUsesGmail dot com ¶

14 years ago

I needed to identify email addresses in a data table that were replicated, so I wrote the array_not_unique() function:

```
<?php

function array_not_unique($raw_array) {
    $dupes = array();
    natcasesort($raw_array);
    reset ($raw_array);

    $old_key    = NULL;
    $old_value    = NULL;
    foreach ($raw_array as $key => $value) {
        if ($value === NULL) { continue; }
        if ($old_value == $value) {
            $dupes[$old_key]    = $old_value;
            $dupes[$key]        = $value;
```

```
        }
        $old_value    = $value;
        $old_key    = $key;
    }
return $dupes;
}

$raw_array     = array();
$raw_array[1]    = 'abc@xyz.com';
$raw_array[2]    = 'def@xyz.com';
$raw_array[3]    = 'ghi@xyz.com';
$raw_array[4]    = 'abc@xyz.com'; // Duplicate

$common_stuff    = array_not_unique($raw_array);
var_dump($common_stuff);
?>
```

up
down
28

mnbayazit ¶

15 years ago

Case insensitive; will keep first encountered value.

```
<?php

function array_iunique($array) {
    $lowered = array_map('strtolower', $array);
    return array_intersect_key($array, array_unique($lowered));
}

?>
```

up
down
3

keneks at gmail dot com ¶

16 years ago

Taking the advantage of array_unique, here is a simple function to check if an array has duplicate values.

It simply compares the number of elements between the original array and the array_uniqued array.

```
<?php

function array_has_duplicates(array $array)
{
    $uniq = array_unique($array);
    return count($uniq) != count($array);
}

?>
```

up
down
1

contact at evoweb dot fr ¶

1 year ago

Here is a solution to make unique values keeping empty values for an array with keys :

```
<?php
```

```
function array_unique_kempty($array) {
    $values = array_unique($array);
    $return = array_combine(array_keys($array), array_fill(0,count($array),null));
    return array_merge($return,$values);
}

$myArray = [
    "test1" => "aaa",
    "test2" => null,
    "test3" => "aaa",
    "test4" => "bbb",
    "test5" => null,
    "test6" => "ccc",
    "test7" => "ddd",
    "test8" => "ccc"
];

echo "<pre>".print_r(array_unique_kempty($myArray),true)."</pre>";

/*
Array
(
    [test1] => aaa
    [test2] =>
    [test3] =>
    [test4] => bbb
    [test5] =>
    [test6] => ccc
    [test7] => ddd
    [test8] =>
)
*/
?>
```

up
down
1
calexandrepcjr at gmail dot com ¶
5 years ago

Following the Ghanshyam Katriya idea, but with an array of objects, where the $key is related to object propriety that you want to filter the uniqueness of array:

```
<?php
function obj_multi_unique($obj, $key = false)
    {
        $totalObjs = count($obj);
        if (is_array($obj) && $totalObjs > 0 && is_object($obj[0]) && ($key && !is_numeric($key)))
{
            for ($i = 0; $i < $totalObjs; $i++) {
                if (isset($obj[$i])) {
                    for ($j = $i + 1; $j < $totalObjs; $j++) {
                        if (isset($obj[$j]) && $obj[$i]->{$key} === $obj[$j]->{$key}) {
                            unset($obj[$j]);
                        }
                    }
                }
            }
            return array_values($obj);
        } else {
```

```
            throw new Exception('Invalid argument or your array of objects is empty');
        }
    }
?>
```

up
down
13

mostafatalebi at rocketmail dot com ¶

8 years ago

If you find the need to get a sorted array without it preserving the keys, use this code which has worked for me:

```
<?php

$array = array("hello", "fine", "good", "fine", "hello", "bye");

$get_sorted_unique_array = array_values(array_unique($array));

?>
```

The above code returns an array which is both unique and sorted from zero.

up
down
20

regeda at inbox dot ru ¶

12 years ago

recursive array unique for multiarrays

```
<?php
function super_unique($array)
{
  $result = array_map("unserialize", array_unique(array_map("serialize", $array)));

  foreach ($result as $key => $value)
  {
    if ( is_array($value) )
    {
      $result[$key] = super_unique($value);
    }
  }

  return $result;
}
?>
```

up
down
3

sashasimkin at gmail dot com ¶

10 years ago

My object unique function:

```
<?php
function object_unique( $obj ){
    $objArray = (array) $obj;

    $objArray = array_intersect_assoc( array_unique( $objArray ), $objArray );

    foreach( $obj as $n => $f ) {
```

```
        if( !array_key_exists( $n, $objArray ) ) unset( $obj->$n );
    }

    return $obj;
}
?>
```

And these code:

```
<?php
class Test{
    public $pr0 = 'string';
    public $pr1 = 'string1';
    public $pr2 = 'string';
    public $pr3 = 'string2';
}

$obj = new Test;

var_dump( object_unique( $obj ) );
?>
```

returns:

```
object(Test)[1]
  public 'pr0' => string 'string' (length=6)
  public 'pr1' => string 'string1' (length=7)
  public 'pr3' => string 'string2' (length=7)
```

up
down
6

agarcia at rsn dot com dot co ¶
16 years ago

This is a script for multi_dimensional arrays

```
<?php
function remove_dup($matriz) {
    $aux_ini=array();
    $entrega=array();
    for($n=0;$n<count($matriz);$n++)
    {
        $aux_ini[]=serialize($matriz[$n]);
    }
    $mat=array_unique($aux_ini);
    for($n=0;$n<count($matriz);$n++)
    {

            $entrega[]=unserialize($mat[$n]);

    }
    return $entrega;
}
?>
```

up
down
3

Ludovico Grossi ¶
7 years ago

[Editor's note: please note that this will not work well with non-scalar values in the array. Array keys can not be arrays themselves, nor streams, resources, etc. Flipping the array causes a change in key-name]

You can do a super fast version of array_unique directly in PHP, even faster than the other solution posted in the comments!

Compared to the built in function it is 20x faster! (2x faster than the solution in the comments).

```
<?php
function superfast_array_unique($array) {
    return array_keys(array_flip($array));
}
?>
```

This works faster for small and big arrays.

up
down
5

quecoder at gmail ¶

14 years ago

another method to get unique values is :

```
<?php
$alpha=array('a','b','c','a','b','d','e','f','f');

$alpha= array_keys(array_count_values($alpha));

print_r($alpha);
?>
```

Output:

```
Array ( [0] => a [1] => b [2] => c [3] => d [4] => e [5] => f )
```

up
down
3

subhrajyoti dot de007 at gmail dot com ¶

4 years ago

Simple and clean way to get duplicate entries removed from a multidimensional array.

```
<?php
          $multi_array = $multi_array [0];
          $multi_array = array_unique($multi_array);
          print_r($multi_array);
?>
```

up
down
2

jusvalceanu - SPAM at SPAM - yahoo dot com ¶

14 years ago

so .... my problem was multidimensional sort.

```
<?php
      $new = array();
      $exclude = array("");
      for ($i = 0; $i<=count($attribs)-1; $i++) {
         if (!in_array(trim($attribs[$i]["price"]) ,$exclude)) { $new[] = $attribs[$i]; $exclude[]
= trim($attribs[$i]["price"]); }
```

```
      }

?>
```

Array $attribs is an array contaning arrays. Each array in the $attrib array consists in multiple fields (ex: name, lenght, price, etc.) to be more simpler in speech think that $attrib is the array resulted by a search sql query done by a visitator on your online shoopping website ... (so ... each array in the $attrib is a product :P) if you want to sort only the uniq results use the above or use this:

```
<?php

   /* Our Array of products */
   $attribs[] = array(
                           "name"         => "Test Product 1",
                           "length"     => "42 cm",
                           "weight"     => "0,5 kg",
                           "price"     => "10 $",
                           "stock"     => "100",
                        );

   $attribs[] = array(
                           "name"         => "Test Product 2",
                           "length"     => "42 cm",
                           "weight"     => "1,5 kg",
                           "price"     => "10 $",
                           "stock"     => "200",
                        );

    /* The nice stuff */

      $new = array();
      $exclude = array("");
      for ($i = 0; $i<=count($attribs)-1; $i++) {
         if (!in_array(trim($attribs[$i]["price"]) ,$exclude)) { $new[] = $attribs[$i]; $exclude[]
= trim($attribs[$i]["price"]); }
      }

      print_r($new); // $new is our sorted array

?>
```

Have fun tweaking this ;)) i know you will ;))

From Romania With Love

up
down
2

webmaster at jukkis dot net ¶

15 years ago

Another way to 'unique column' an array, in this case an array of objects:
Keep the desired unique column values in a static array inside the callback function for array_filter.

Example:

```
<?php
/* example object */
class myObj {
```

```
  public $id;
  public $value;
  function __construct( $id, $value ) {
    $this->id = $id;
    $this->value = $value;
  }
}

/* callback function */
function uniquecol( $obj ) {
  static $idlist = array();

  if ( in_array( $obj->id, $idlist ) )
    return false;

  $idlist[] = $obj->id;
  return true;
}

/* a couple of arrays with second array having an element with same id as the first */
$list  = array( new myObj( 1, 1  ), new myObj( 2, 100 ) );
$list2 = array( new myObj( 1, 10 ), new myObj( 3, 100 ) );
$list3 = array_merge( $list, $list2 );

$unique = array_filter( $list3, 'uniquecol' );
print_r( $list3 );
print_r( $unique );

?>

In addition, use array_merge( $unique ) to reindex.
```

up
down
2
amri [at t] dhstudio dot eu ¶
12 years ago

```
I searched how to show only the de-duplicate elements from array, but failed.
Here is my solution:

<?php
function arrayUniqueElements($array)
{
return array_unique(array_diff_assoc($array1,array_unique($array1)));
};
?>

Example:
<?php
$arr1 = array('foo', 'bar', 'xyzzy', '&', 'xyzzy',
'baz', 'bat', '|', 'xyzzy', 'plugh',
'xyzzy', 'foobar', '|', 'plonk', 'xyzzy',
'apples', '&', 'xyzzy', 'oranges', 'xyzzy',
'pears','foobar');

$result=arrayUniqueElements($arr1);
print_r($result);exit;
?>
```

```
Output:

Array
(
[4] => xyzzy
[12] => |
[16] => &
[21] => foobar
)
```

up
down
1

csaba at alum dot mit dot edu ¶

18 years ago

The following is an efficient, adaptable implementation of array_unique which always retains the first key having a given value:

```
<?php
function array_unique2(&$aray) {
    $aHash = array();
    foreach ($aray as $key => &$val) if (@$aHash[$val]++) unset ($aray[$key]);
}
?>
```

It is also adaptable to multi dimensional arrays. For example, if your array is a sequence of (multidimensional) points, then in place of @$aHash[$val]++ you could use
@$aHash[implode("X",$val)]++
If you want to not have holes in your array, you can do an array_merge($aray) at the end.

Csaba Gabor

up
down
0

free dot smilesrg at gmail dot com ¶

15 days ago

```
$a = new StdClass();
$b = new StdClass();

var_dump(array_unique([$a, $b, $b, $a], SORT_REGULAR));
//array(1) {
//  [0]=>
//    object(stdClass)#1 (0) {
//  }
//}

$a->name = 'One';
$b->name = 'Two';

var_dump(array_unique([$a, $b, $b, $a], SORT_REGULAR));

//array(2) {
//  [0]=>
//  object(stdClass)#1 (1) {
//    ["name"]=>
//    string(3) "One"
//  }
//  [1]=>
//  object(stdClass)#2 (1) {
```

```
//    ["name"]=>
//    string(3) "Two"
//  }
//}
```

up
down
0

Sbastien ¶

5 months ago

Because of PHP comparaisons modalities, you can never distinguish null from others falsy values.
Note the absorbing nature of true and false booleans in mix types array.

```
<?php

$a = [true, false, null, '', '0', '123', 0, 123];
foreach (['SORT_REGULAR', 'SORT_NUMERIC', 'SORT_STRING', 'SORT_LOCALE_STRING'] as $flag) {
    $a_new = array_unique($a, constant($flag));
    echo "{$flag} ==> ";
    var_dump($a_new);
}

/*

Gives :

SORT_REGULAR ==> array(2) {
  [0]=> bool(true)
  [1]=> bool(false)
}
SORT_NUMERIC ==> array(3) {
  [0]=> bool(true)
  [1]=> bool(false)
  [5]=> string(3) "123"
}
SORT_STRING ==> array(4) {
  [0]=> bool(true)
  [1]=> bool(false)
  [4]=> string(1) "0"
  [5]=> string(3) "123"
}
SORT_LOCALE_STRING ==> array(4) {
  [0]=> bool(true)
  [1]=> bool(false)
  [4]=> string(1) "0"
  [5]=> string(3) "123"
}

*/
```

up
down
0

zoolyka at gmail dot com ¶

6 years ago

I found the simplest way to "unique" multidimensional arrays as follows:

```
<?php

$array = array(
```

```
    'a' => array(1, 2),
    'b' => array(1, 2),
    'c' => array(2, 2),
    'd' => array(2, 1),
    'e' => array(1, 1),
);

$array = array_map('json_encode', $array);
$array = array_unique($array);
$array = array_map('json_decode', $array);

print_r($array);

?>
```

As you can see "b" will be removed without any errors or notices.

up
down
0

dirk dot avery a t gmail ¶

13 years ago

Although array_unique is not intended to work with multi-dimensional arrays, it does on 5.2.9. However, it does not for 5.2.5. Beware.

up
down
0

Dorphalsig ¶

14 years ago

I had a problem with array_unique and multidimensional arrays ... Maybe there's a better way to do this, but this will work for any dimensional arrays.

```
<?php
function arrayUnique($myArray)
{
    if(!is_array($myArray))
           return $myArray;

    foreach ($myArray as &$myvalue){
        $myvalue=serialize($myvalue);
    }

    $myArray=array_unique($myArray);

    foreach ($myArray as &$myvalue){
        $myvalue=unserialize($myvalue);
    }

    return $myArray;

}
?>
```

up
down
0

PHP Expert ¶

14 years ago

Case insensitive for PHP v4.x and up.

```
<?php

function in_iarray($str, $a) {
    foreach ($a as $v) {
        if (strcasecmp($str, $v) == 0) {
            return true;
        }
    }
    return false;
}

function array_iunique($a) {
    $n = array();
    foreach ($a as $k => $v) {
        if (!in_iarray($v, $n)) {
            $n[$k]=$v;
        }
    }
    return $n;
}

$input = array("aAa","bBb","cCc","AaA","ccC","ccc","CCC","bBB","AAA","XXX");
$result = array_iunique($input);
print_r($result);

/*
Array
(
    [0] => aAa
    [1] => bBb
    [2] => cCc
    [9] => XXX
)
*/
?>
```

up
down
0

geuis dot teses at gmail dot com ¶

16 years ago

Here's the shortest line of code I could find/create to remove all duplicate entries from an array and then reindex the keys.

```
<?php

// Fruits, vegetables, and other food:
$var = array('apple','banana','carrot','cat','dog','egg','eggplant','fish');

$var = array_values(array_unique($var));
?>
```

up
down
0

memandeemail at gmail dot com ¶

16 years ago

Problem:
I have loaded an array with the results of a database query. The Fields are 'FirstName' and 'LastName'.

```
I would like to find a way to contactenate the two
fields, and then return only unique values for the
array.  For example, if the database query returns
three instances of a record with the FirstName John
and the LastName Smith in two distinct fields, I would
like to build a new array that would contain all the
original fields, but with John Smith in it only once.
Thanks for: Colin Campbell

Solution:

<?php
/**
* The same thing than implode function, but return the keys so
*
* <code>
* $_GET = array('id' => '4587','with' => 'key');
* ...
* echo shared::implode_with_key('&',$_GET,'='); // Resultado: id=4587&with=key
* ...
* </code>
*
* @param string $glue Oque colocar entre as chave => valor
* @param array $pieces Valores
* @param string $hifen Separar chave da array do valor
* @return string
* @author memandeemail at gmail dot com
*/
function implode_with_key($glue = null, $pieces, $hifen = ',') {
  $return = null;
  foreach ($pieces as $tk => $tv) $return .= $glue.$tk.$hifen.$tv;
  return substr($return,1);
}

/**
* Return unique values from a tree of values
*
* @param array $array_tree
* @return array
* @author memandeemail at gmail dot com
*/
function array_unique_tree($array_tree) {
  $will_return = array(); $vtemp = array();
  foreach ($array_tree as $tkey => $tvalue) $vtemp[$tkey] = implode_with_key('&',$tvalue,'=');
  foreach (array_keys(array_unique($vtemp)) as $tvalue) $will_return[$tvalue] =
$array_tree[$tvalue];
  return $will_return;
}

$problem = array_fill(0,3,
array('FirstName' => 'John', 'LastName' => 'Smith')
);

$problem[] = array('FirstName' => 'Davi', 'LastName' => 'S. Mesquita');
$problem[] = array('FirstName' => 'John', 'LastName' => 'Tom');

print_r($problem);
```