Focus search box

array_fill_keys »
« array_diff_ukey

- 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_diff

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

array_diff — Calcula la diferencia entre arrays

Descripción ¶

array_diff(array `$array1`, array `$array2`, array `$...` = ?): array

Compara `array1` con uno o más arrays y devuelve los valores de `array1` que no estén presentes en ninguno de los otros arrays.

Parámetros ¶

`array1`

El array a comparar

`array2`

Un array con el que comparar

`...`

Más arrays con los que comparar

Valores devueltos ¶

Devuelve un array que contiene todas las entradas de `array1` que no están presentes en ninguna de los otros arrays.

Ejemplos ¶

Ejemplo #1 Ejemplo de array_diff()

```
<?php
$array1    = array("a" => "green", "red", "blue", "red");
$array2    = array("b" => "green", "yellow", "red");
$resultado = array_diff($array1, $array2);
```

```
print_r($resultado);
?>
```

Todas las múltiples coincidencias en *$array1* serán tratadas de la misma manera. Esta será la salida:

```
Array
(
    [1] => blue
)
```

Notas ¶

> **Nota**:
>
> Dos elementos son considerandos iguales si y sólo si `(string) $elem1 === (string) $elem2`. En otras palabras: cuando la representación de string es la misma.
>
> **Nota**:
>
> Esta función sólo comprueba una dimensión de un array n-dimensional. Por supuesto, se pueden comprobar arrays de más dimensiones usando `array_diff_assoc($array1[0], $array2[0]);`.

Ver también ¶

- array_diff_assoc() - Calcula la diferencia entre arrays con un chequeo adicional de índices
- array_intersect() - Calcula la intersección de arrays
- array_intersect_assoc() - Calcula la intersección de arrays con un chequeo adicional de índices

+ add a note

User Contributed Notes 27 notes

up
down
227
nilsandre at gmx dot de ¶
15 years ago

```
Again, the function's description is misleading right now. I sought a function, which
(mathematically) computes A - B, or, written differently, A \ B. Or, again in other words, suppose

A := {a1, ..., an} and B:= {a1, b1, ... , bm}

=> array_diff(A,B) = {a2, ..., an}

array_diff(A,B) returns all elements from A, which are not elements of B (= A without B).

You should include this in the documentation more precisely, I think.
```

up
down
66
Anonymous ¶
16 years ago

```
array_diff provides a handy way of deleting array elements by their value, without having to unset
it by key, through a lengthy foreach loop and then having to rekey the array.

<?php

//pass value you wish to delete and the array to delete from
```

```
function array_delete( $value, $array)
{
    $array = array_diff( $array, array($value) );
    return $array;
}
?>
```

up
down
34

james dot PLZNOSPAM at bush dot cc ¶
5 years ago

If you want a simple way to show values that are in either array, but not both, you can use this:

```
<?php
function arrayDiff($A, $B) {
    $intersect = array_intersect($A, $B);
    return array_merge(array_diff($A, $intersect), array_diff($B, $intersect));
}
?>
```

If you want to account for keys, use array_diff_assoc() instead; and if you want to remove empty values, use array_filter().

up
down
22

firegun at terra dot com dot br ¶
13 years ago

Hello guys,

I´ve been looking for a array_diff that works with recursive arrays, I´ve tried the ottodenn at gmail dot com function but to my case it doesn´t worked as expected, so I made my own. I´ve haven´t tested this extensively, but I´ll explain my scenario, and this works great at that case :D

We got 2 arrays like these:

```
<?php
$aArray1['marcie'] = array('banana' => 1, 'orange' => 1, 'pasta' => 1);
$aArray1['kenji'] = array('apple' => 1, 'pie' => 1, 'pasta' => 1);

$aArray2['marcie'] = array('banana' => 1, 'orange' => 1);
?>
```

As array_diff, this function returns all the items that is in aArray1 and IS NOT at aArray2, so the result we should expect is:

```
<?php
$aDiff['marcie'] = array('pasta' => 1);
$aDiff['kenji'] = array('apple' => 1, 'pie' => 1, 'pasta' => 1);
?>
```

Ok, now some comments about this function:
- Different from the PHP array_diff, this function DON´T uses the === operator, but the ==, so 0 is equal to '0' or false, but this can be changed with no impacts.
- This function checks the keys of the arrays, array_diff only compares the values.

I realy hopes that this could help some1 as I´ve been helped a lot with some users experiences. (Just please double check if it would work for your case, as I sad I just tested to a scenario like the one I exposed)

```
<?php
function arrayRecursiveDiff($aArray1, $aArray2) {
    $aReturn = array();

    foreach ($aArray1 as $mKey => $mValue) {
        if (array_key_exists($mKey, $aArray2)) {
            if (is_array($mValue)) {
                $aRecursiveDiff = arrayRecursiveDiff($mValue, $aArray2[$mKey]);
                if (count($aRecursiveDiff)) { $aReturn[$mKey] = $aRecursiveDiff; }
            } else {
                if ($mValue != $aArray2[$mKey]) {
                    $aReturn[$mKey] = $mValue;
                }
            }
        } else {
            $aReturn[$mKey] = $mValue;
        }
    }

    return $aReturn;
}
?>
```

up
down
40

Jeppe Utzon ¶
10 years ago

If you just need to know if two arrays' values are exactly the same (regardless of keys and order), then instead of using array_diff, this is a simple method:

```
<?php

function identical_values( $arrayA , $arrayB ) {

    sort( $arrayA );
    sort( $arrayB );

    return $arrayA == $arrayB;
}

// Examples:

$array1 = array( "red" , "green" , "blue" );
$array2 = array( "green" , "red" , "blue" );
$array3 = array( "red" , "green" , "blue" , "yellow" );
$array4 = array( "red" , "yellow" , "blue" );
$array5 = array( "x" => "red" , "y" =>  "green" , "z" => "blue" );

identical_values( $array1 , $array2 );  // true
identical_values( $array1 , $array3 );  // false
identical_values( $array1 , $array4 );  // false
identical_values( $array1 , $array5 );  // true

?>
```

The function returns true only if the two arrays contain the same number of values and each value in one array has an exact duplicate in the other array. Everything else will return false.

my alternative method for evaluating if two arrays contain (all) identical values:

```
<?php

sort($a); $sort(b); return $a == $b;

?>
```

may be slightly faster (10-20%) than this array_diff method:

```
<?php

return ( count( $a ) == count( $b ) && !array_diff( $a , $b ) ? true : false );

?>
```

but only when the two arrays contain the same number of values and then only in some cases. Otherwise the latter method will be radically faster due to the use of a count() test before the array_diff().

Also, if the two arrays contain a different number of values, then which method is faster will depend on whether both arrays need to be sorted or not. Two times sort() is a bit slower than one time array_diff(), but if one of the arrays have already been sorted, then you only have to sort the other array and this will be almost twice as fast as array_diff().

Basically: 2 x sort() is slower than 1 x array_diff() is slower than 1 x sort().

up
down
26

SeanECoates at !donotspam!yahoo dot ca ¶

21 years ago

I just came upon a really good use for array_diff(). When reading a dir(opendir;readdir), I _rarely_ want "." or ".." to be in the array of files I'm creating. Here's a simple way to remove them:

```
<?php
$someFiles = array();
$dp = opendir("/some/dir");
while($someFiles[] = readdir($dp));
closedir($dp);

$removeDirs = array(".","..");
$someFiles = array_diff($someFiles, $removeDirs);

foreach($someFiles AS $thisFile) echo $thisFile."\n";
?>
```

S

up
down
4

Al Amin Chayan (mail at chayan dot me) ¶

2 years ago

```
<?php
/**
   * Check If An Array Is A Subset Of Another Array
   *
```

```
    * @param  array $subset
    * @param  array $set
    * @return bool
    */
function is_subset(array $subset, array $set): bool {
     return (bool)!array_diff($subset, $set);
}

$u = [1, 5, 6, 8, 10];
$a = [1, 5];
$b = [6, 7];

var_dump(is_subset($a, $u)); // true
var_dump(is_subset($b, $u)); // false
```

up
down
2

Prakashgun ¶
4 years ago

If duplicate value comes in the first array, that will be also included. See in the output "blue" comes twice.

```
<?php

$array1 = array("a" => "green", "red", "blue", "red", "blue");
$array2 = array("b" => "green", "yellow", "red");
$result = array_diff($array1, $array2);

print_r($result);

//Output
//Array ( [1] => blue [3] => blue )
```

up
down
13

merlyn dot tgz at gmail dot com ¶
10 years ago

There is more fast implementation of array_diff, but with some limitations. If you need compare two arrays of integers or strings you can use such function:

```
    public static function arrayDiffEmulation($arrayFrom, $arrayAgainst)
    {
        $arrayAgainst = array_flip($arrayAgainst);

        foreach ($arrayFrom as $key => $value) {
            if(isset($arrayAgainst[$value])) {
                unset($arrayFrom[$key]);
            }
        }

        return $arrayFrom;
    }
```

It is ~10x faster than array_diff

```
php > $t = microtime(true);$a = range(0,25000); $b = range(15000,500000); $c = array_diff($a, $b);echo microtime(true) - $t;
4.4335179328918
```

```
php > $t = microtime(true);$a = range(0,25000); $b = range(15000,500000); $c =
arrayDiffEmulation($a, $b);echo microtime(true) - $t;
0.37219095230103
```

up
down
3
wes dot melton at gmail dot com ¶
5 years ago

It's important to note that array_diff() is NOT a fast or memory-efficient function on larger arrays.

In my experience, when I find myself running array_diff() on larger arrays (50+ k/v/pairs) I almost always realize that I'm working the problem from the wrong angle.

Typically, when reworking the problem to not require array_diff(), especially on bigger datasets, I find significant performance improvements and optimizations.

up
down
7
Anonymous ¶
7 years ago

I always wanted something like this to avoid listing all the files and folders you want to exclude in a project directory.

```
function array_preg_diff($a, $p) {
    foreach ($a as $i => $v)
        if (preg_match($p, $v))
            unset($a[$i]);
    return $a;
}

$relevantFiles = array_preg_diff(scandir('somedir'), '/^\./');
```

instead of

```
$relevantFiles = array_diff(scandir('somedir'), array('.', '..', '.idea', '.project));
```

up
down
3
merlinyoda at dorproject dot net ¶
14 years ago

As touched on in kitchin's comment of 19-Jun-2007 03:49 and nilsandre at gmx dot de's comment of 17-Jul-2007 10:45, array_diff's behavior may be counter-intuitive if you aren't thinking in terms of set theory.

array_diff() returns a *mathematical* difference (a.k.a. subtraction) of elements in array A that are in array B and *not* what elements are different between the arrays (i.e. those that elements that are in either A or B but aren't in both A and B).

Drawing one of those Ven diagrams or Euler diagrams may help with visualization...

As far as a function for returning what you may be expecting, here's one:

```
<?php
function array_xor ($array_a, $array_b) {
    $union_array = array_merge($array_a, $array_b);
    $intersect_array = array_intersect($array_a, $array_b);
    return array_diff($union_array, $intersect_array)
```

```
}
?>
```

up
down
5

Anonymous ¶
18 years ago

From the page:
Note: Please note that this function only checks one dimension of a n-dimensional array. Of course you can check deeper dimensions by using array_diff($array1[0], $array2[0]);

I've found a way to bypass that. I had 2 arrays made of arrays.
I wanted to extract from the first array all the arrays not found in the second array. So I used the serialize() function:

```
<?php
function my_serialize(&$arr,$pos){
  $arr = serialize($arr);
}

function my_unserialize(&$arr,$pos){
  $arr = unserialize($arr);
}

//make a copy
$first_array_s = $first_array;
$second_array_s = $second_array;

// serialize all sub-arrays
array_walk($first_array_s,'my_serialize');
array_walk($second_array_s,'my_serialize');

// array_diff the serialized versions
$diff = array_diff($first_array_s,$second_array_s);

// unserialize the result
array_walk($diff,'my_unserialize');

// you've got it!
print_r($diff);
?>
```

up
down
3

emeka at echeruo at gmail dot com ¶
6 years ago

Resubmitting... the update for takes into account comparison issues

Computes the difference of all the arrays

```
<?php

/**
* array_diffs – Computes the difference of all the arrays
*
* @param array
*
*    array1 - The array to compare from and against
```

```
 *    array2 - The array to compare from and against
 *    array(n) - More arrays to compare from and against
 *
 * @return array Returns all the arrays that do contains entries that cannot be matched in any of
 the arrays.
 */

 function array_diffs() {
     $count = func_num_args();
     if($count < 2) {
         trigger_error('Must provide at least 2 arrays for comparison.');
     }
     $check = array();
     $out = array();
      //resolve comparison issues in PHP
     $func = function($a, $b) {
         if(gettype($a) == 'integer' && gettype($b['value']) == 'double' || gettype($a) == 'double'
 && gettype($b['value']) == 'integer') {
             if(gettype($a) == 'integer') {
                 if((double) $a == $b['value']) {
                     return true;
                 }
             } else {
                 if((double) $b['value'] == $a) {
                     return true;
                 }
             }
         } elseif(gettype($a) == 'double' && gettype($b['value']) == 'double') {
             $epsilon = 0.00001;
             if(abs($a - $b['value']) < $epsilon) {
                 return true;
             }
         } else {
             if($a == $b['value']) {
                 return true;
             }
         }
         return false;
     };
     for ($i = 0; $i < $count; $i++) {
         if(!is_array(func_get_arg($i))) {
             trigger_error('Parameters must be passed as arrays.');
         }
         foreach(func_get_arg($i) as $key => $value) {
             if(is_numeric($key) && is_string($value)) {
                 if(array_key_exists($value, $check) && $func($value, $check[$value])) {
                     $check[$value]['count'] = $check[$value]['count'] + 1;
                 } else {
                     $check[$value]['value'] = $value;
                     $check[$value]['count'] = 1;
                 }
             } elseif(is_numeric($key) && (is_bool($value) || is_null($value) || is_numeric($value)
 || is_object($value) || is_resource($value))) {
                 $update = false;
                 foreach($check as $check_key => $check_value) {
                     if(is_numeric($key) && (is_bool($check_value['value']) ||
 is_null($check_value['value']) || is_numeric($check_value['value']) ||
 is_object($check_value['value']) || is_resource($check_value['value']))) {
```

```
                        if($func($value, $check_value)) {
                            $update = true;
                            $check[$check_key]['count'] = $check[$check_key]['count'] + 1;
                        }
                    }
                }
                if(!$update) {
                    $check[] = array('value' => $value, 'count' => 1);
                }
            } else {
                if(array_key_exists($key, $check) && $func($value, $check[$key])) {
                    $check[$key]['count'] = $check[$key]['count'] + 1;
                } else {
                    $check[$key]['value'] = $value;
                    $check[$key]['count'] = 1;
                }
            }
        }
    }
    foreach($check as $check_key => $check_value) {
        if($check_value['count'] == 1) {
            for ($i = 0; $i < $count; $i++) {
                foreach(func_get_arg($i) as $key => $value) {
                    if(is_numeric($key) && is_string($value)) {
                        if($value == (string) $check_key) {
                            $out[$i][$key] = $value;
                        }
                    } elseif(is_numeric($key) && (is_bool($value) || is_null($value) ||
is_numeric($value) || is_object($value) || is_resource($value))) {
                        if(is_numeric($key) && (is_bool($check_value['value']) ||
is_null($check_value['value']) || is_numeric($check_value['value']) ||
is_object($check_value['value']) || is_resource($check_value['value']))) {
                            if($check_value['value'] == $value) {
                                $out[$i][$key] = $value;
                            }
                        }
                    } else {
                        if($key == $check_key) {
                            $out[$i][$key] = $value;
                        }
                    }
                }
            }
        }
    }
    return $out;
}

?>
```

up
down
2

Tim Trefren ¶

14 years ago

Here's a little wrapper for array_diff - I found myself needing to iterate through the edited array, and I didn't need to original keys for anything.

```
<?php
```

```
function arrayDiff($array1, $array2){
    # This wrapper for array_diff rekeys the array returned
    $valid_array = array_diff($array1,$array2);

    # reinstantiate $array1 variable
    $array1 = array();

    # loop through the validated array and move elements to $array1
    # this is necessary because the array_diff function returns arrays that retain their original
keys
    foreach ($valid_array as $valid){
        $array1[] = $valid;
        }
    return $array1;
    }
?>
```

up
down
2

eugeny dot yakimovitch at gmail dot com ¶

11 years ago

Note that array_diff is not equivalent to

```
<?php
function fullArrayDiff($left, $right)
{
    return array_diff(array_merge($left, $right), array_intersect($left, $right));
}
?>
```

since it is a set-theoretical complement as in

http://en.wikipedia.org/wiki/Complement_(set_theory)

up
down
1

emeka dot echeruo at gmail dot com ¶

6 years ago

Resubmitting... the update for takes into account comparison issues
Computes the difference of all the arrays

```
<?php

/**
* array_diffs â€” Computes the difference of all the arrays
*
* @param array
*
*    array1 - The array to compare from and against
*    array2 - The array to compare from and against
*    array(n) - More arrays to compare from and against
*
* @return array Returns all the arrays that do contains entries that cannot be matched in any of
the arrays.
*/

function array_diffs() {
    $count = func_num_args();
    if($count < 2) {
```

```
        trigger_error('Must provide at least 2 arrays for comparison.');
    }
    $check = array();
    $out = array();
     //resolve comparison issue
    $func = function($a, $b) {
        $dbl = function($i, $d) {
            $e = 0.00001;
            if(abs($i-$d) < $e) {
                return true;
            }
            return false;
        };
        if((gettype($a) == 'integer' && gettype($b['value']) == 'double') || (gettype($a) ==
'double' && gettype($b['value']) == 'integer')) {
            if((gettype($a) == 'integer' && $dbl((double) $a, $b['value'])) ||
(gettype($b['value']) == 'integer' && $dbl((double) $b['value'], $a))) {
                return true;
            }
        } elseif((gettype($a) == 'double') && (gettype($b['value']) == 'double')) {
            return $dbl($a,$b['value']);
        } elseif($a == $b['value']) {
            return true;
        }
        return false;
    };
    for($i = 0; $i < $count; $i++) {
        if(!is_array(func_get_arg($i))) {
            trigger_error('Parameters must be passed as arrays.');
        }
        foreach(func_get_arg($i) as $key => $value) {
            if(is_numeric($key) && is_string($value)) {
                if(array_key_exists($value, $check) && $func($value, $check[$value])) {
                    $check[$value]['count'] = $check[$value]['count'] + 1;
                } else {
                    $check[$value]['value'] = $value;
                    $check[$value]['count'] = 1;
                }
            } elseif(is_numeric($key) && (is_bool($value) || is_null($value) || is_numeric($value)
|| is_object($value) || is_resource($value))) {
                $update = false;
                foreach($check as $check_key => $check_value) {
                    if(is_numeric($key) && (is_bool($check_value['value']) ||
is_null($check_value['value']) || is_numeric($check_value['value']) ||
is_object($check_value['value']) || is_resource($check_value['value'])) && $func($value,
$check_value)) {
                        $update = true;
                        $check[$check_key]['count'] = $check[$check_key]['count'] + 1;
                    }
                }
                if(!$update) {
                    $check[] = array('value' => $value, 'count' => 1);
                }
            } else {
                if(array_key_exists($key, $check) && $func($value, $check[$key])) {
                    $check[$key]['count'] = $check[$key]['count'] + 1;
                } else {
                    $check[$key]['value'] = $value;
```

```
                    $check[$key]['count'] = 1;
                }
            }
        }
    }
    foreach($check as $check_key => $check_value) {
        if($check_value['count'] == 1) {
           for ($i = 0; $i < $count; $i++) {
                foreach(func_get_arg($i) as $key => $value) {
                    if(is_numeric($key) && is_string($value) && ($value == (string) $check_key)) {
                        $out[$i][$key] = $value;
                    } elseif(is_numeric($key) && ($check_value['value'] == $value)) {
                        $out[$i][$key] = $value;
                    } elseif(is_string($key) && ($check_value['value'] == $value)) {
                        $out[$i][$key] = $value;
                    }
                }
            }
        }
    }
    return $out;
}

?>
```

up
down
3

Simon Riget at paragi.dk ¶
16 years ago

A simple multidimentional key aware array_diff function.

```
<?php
function arr_diff($a1,$a2){
  foreach($a1 as $k=>$v){
    unset($dv);
    if(is_int($k)){
      // Compare values
      if(array_search($v,$a2)===false) $dv=$v;
      else if(is_array($v)) $dv=arr_diff($v,$a2[$k]);
      if($dv) $diff[]=$dv;
    }else{
      // Compare noninteger keys
      if(!$a2[$k]) $dv=$v;
      else if(is_array($v)) $dv=arr_diff($v,$a2[$k]);
      if($dv) $diff[$k]=$dv;
    }
  }
  return $diff;
}
?>
```

This function meets my immidiate needs but I'm shure it can be improved.

up
down
2

javierchinapequeno at yahoo dot es ¶
11 years ago

Hi, I´d like to give a piece of advice to all who need to use this function to compare two arrays that have a great quantity of elements. You should sort both arrays first before comparing, it will work faster.
Thanks

up
down
3

vojtech dot hordejcuk at gmail dot com ¶

13 years ago

Based on one lad's code, I created following function for creating something like HTML diff. I hope it will be useful.

```
<?php
private function diff ($old, $new)
{
  $old = preg_replace ('/ +/', ' ', $old);
  $new = preg_replace ('/ +/', ' ', $new);

  $lo = explode ("\n", trim ($old) . "\n");
  $ln = explode ("\n", trim ($new) . "\n");
  $size = max (count ($lo), count ($ln));

  $equ = array_intersect ($lo, $ln);
  $ins = array_diff ($ln, $lo);
  $del = array_diff ($lo, $ln);

  $out = '';

  for ($i = 0; $i < $size; $i++)
  {
    if (isset ($del [$i]))
    {
      $out .= '<p><del>' . $del [$i] . '</del></p>';
    }

    if (isset ($equ [$i]))
    {
      $out .= '<p>' . $equ [$i] . '</p>';
    }

    if (isset ($ins [$i]))
    {
      $out .= '<p><ins>' . $ins [$i] . '</ins></p>';
    }
  }

  return $out;
}
?>
```

up
down
2

wrey75 at gmail dot com ¶

7 years ago

The difference is made only on the first level. If you want compare 2 arrays, you can use the code available at https://gist.github.com/wrey75/c631f6fe9c975354aec7 (including a class with an function to patch the array)

Here the basic function:

```
function my_array_diff($arr1, $arr2) {
        $diff = array();

        // Check the similarities
        foreach( $arr1 as $k1=>$v1 ){
            if( isset( $arr2[$k1]) ){
                $v2 = $arr2[$k1];
                if( is_array($v1) && is_array($v2) ){
                    // 2 arrays: just go further...
                    // .. and explain it's an update!
                    $changes = self::diff($v1, $v2);
                    if( count($changes) > 0 ){
                        // If we have no change, simply ignore
                        $diff[$k1] = array('upd' => $changes);
                    }
                    unset($arr2[$k1]); // don't forget
                }
                else if( $v2 === $v1 ){
                    // unset the value on the second array
                    // for the "surplus"
                    unset( $arr2[$k1] );
                }
                else {
                    // Don't mind if arrays or not.
                    $diff[$k1] = array( 'old' => $v1, 'new'=>$v2 );
                    unset( $arr2[$k1] );
                }
            }
            else {
                // remove information
                $diff[$k1] = array( 'old' => $v1 );
            }
        }

        // Now, check for new stuff in $arr2
        reset( $arr2 ); // Don't argue it's unnecessary (even I believe you)
        foreach( $arr2 as $k=>$v ){
            // OK, it is quite stupid my friend
            $diff[$k] = array( 'new' => $v );
        }
        return $diff;
    }
```

up
down
2

Anonymous ¶

13 years ago

Hi!
I tried hard to find a solution to a problem I'm going to explain here, and after have read all the array functions and possibilities, I had to create what I think should exist on next PHP releases.

What I needed, it's some kind of Difference, but working with two arrays and modifying them at time, not returning an array as a result with the diference itself.

So, as an example:

A = 1,2,3
B = 2,3,4

should NOT be:

C = 1,4

but:

A = 1
B = 4

so basically, I wanted to delete coincidences on both arrays.

Now, I've some actions to do, and I know wich one I've to do with the values from one array or another.
With the normal DIFF I can't, because if I've an array like C=1,4, I dont know if I've to do the Action_A with 1 or with 4, but I really know that everything in A, will go to the Action_A and everithing in B, will go to Action_B. So same happens with 4, don't know wich action to apply...

So I created this:

```
<?php
function array_diff_ORG_NEW(&$org, &$new, $type='VALUES'){
    switch($type){
        case 'VALUES':
            $int = array_values(array_intersect($org, $new)); //C = A ^ B
            $org = array_values(array_diff($org, $int)); //A' = A - C
            $new= array_values(array_diff($new, $int)); //B' = B - C
            break;
        case 'KEYS':
            $int = array_values(array_intersect_key($org, $new)); //C = A ^ B
            $org = array_values(array_diff_key($org, $int)); //A' = A - C
            $new= array_values(array_diff_key($new, $int)); //B' = B - C
            break;
    }
}
?>
```

This cute, works by reference, and modifies the arrays deleting coincidences on both, and leaving intact the non coincidences.

So a call to this will be somethin' like:

```
<?php
$original = array(1,2,3);
$new = array(2,3,4);

array_diff_ORG_NEW($original, $new, 'VALUES');
?>
```

And HERE, I'll have my arrays as I wanted:

$original = 1
$new = 4

Now, why I use it precisely?

Imagine you've some "Events" and some users you select when create the event, can "see" this event you create. So you "share" the event with some users. Ok?

Imagine you created and Event_A, and shared with users 1,2,3.

Now you want to modify the event, and you decide to modify the users to share it. Imagine you change it to users 2,3,4.

(numbers are users ID).

So you can manage when you are going to modify, to have an array with the IDs in DDBB ($original), and then, have another array with ID's corresponding to the users to share after modifying ($new). Wich ones you've to DELETE from DDBB, and wich ones do you've to INSERT?

If you do a simple difference or somehow, you get somethin' like C=1,4.
You have no clue on wich one you've to insert or delete.

But on this way, you can know it, and that's why:

- What keeps on $original, it's somethin not existing in $new at the beggining. So you know that all what you've inside $original, have to be deleted from DDBB because what you did in the modifying process, it's to unselect those users keeping in $original.
- What keeps on $new, it's something not existing in $original at the beggining. Wich means that in the modifying process you added some new users. And those have to be inserted in DDBB. So, everything keeping inside $new, have to be inserted in the DDBB.

Conclusion:

- Remaining in $original --> delete from DB.
- Remaining in $new --> insert into DB.

And that's all!

I hope you find it useful, and I encourage PHP "makers", to add in a not distant future, somethin' like this one natively, because I'm shure that I'm not the first one needing something like this.

Best regards all,

Light.

up
down
1

Viking Coder ¶

16 years ago

To anybody wanting a double-sided array_diff - mentioned by rudigier at noxx dot at. Remember, array_diff gives you everything in the first array that isn't in the subsequent arrays.

```
$array1=array('blue','red','green');
$array2=array('blue','yellow','green');

array_merge(array_diff($array1, $array2),array_diff($array2, $array1));

Result
------
Array
(
    [0] => red
```

```
    [1] => yellow
)
```

up
down
0

Anonymous ¶

7 years ago

If you're not getting a count(array_diff($a1,$a2))>0 with something similar to the following arrays should use the php.net/array_diff_assoc function instead.

```
$a1 = Array
(
    [0] => id
    [1] => id_1
    [2] => id_2
)

$a2 = Array
(
    [0] => id
    [1] => id_2
    [2] => id_1
)
```

up
down
0

gilthans at NOgmailSPAM dot com ¶

15 years ago

I needed a function to only remove the element the amount of times he appears in the second array. In other words, if you have Array(1, 1, 2) and Array(1), the return value should be Array(1, 2). So I built this function right here:

```
<?php
function array_diff_once(){
    if(($args = func_num_args()) < 2)
        return false;
    $arr1 = func_get_arg(0);
    $arr2 = func_get_arg(1);
    if(!is_array($arr1) || !is_array($arr2))
        return false;
    foreach($arr2 as $remove){
        foreach($arr1 as $k=>$v){
            if((string)$v === (string)$remove){ //NOTE: if you need the diff to be STRICT, remove
both the '(string)'s
                unset($arr1[$k]);
                break; //That's pretty much the only difference from the real array_diff :P
            }
        }
    }
    //Handle more than 2 arguments
    $c = $args;
    while($c > 2){
        $c--;
        $arr1 = array_diff_once($arr1, func_get_arg($args-$c+1));
    }
    return $arr1;
}
$arr1 = Array("blue", "four"=>4, "color"=>"red", "blue", "green", "green", "name"=>"jon",
```

```
"green");
$arr2 = Array("4", "red", "blue", "green");
print_r(array_diff_once($arr1, $arr2));
?>
This prints:
Array ( [1] => blue [3] => green [name] => jon [4] => green )
```

Note that it removes the elements left to right, opposite to what you might expect; in my case the order of elements had no importance. Fixing that would require a small variation.

up
down
0

j dot j dot d dot mol at ewi dot tudelft dot nl ¶

17 years ago

Here is some code to take the difference of two arrays. It allows custom modifications like prefixing with a certain string (as shown) or custom compare functions.

```
<?php
// returns all elements in $all which are not in $used in O(n log n) time.
  // elements from $all are prefixed with $prefix_all.
  // elements from $used are prefixed with $prefix_used.
  function filter_unused( $all, $used, $prefix_all = "", $prefix_used = "" ) {
      $unused = array();

      // prefixes are not needed for sorting
      sort( $all );
      sort( $used );

      $a = 0;
      $u = 0;

      $maxa = sizeof($all)-1;
      $maxu = sizeof($used)-1;

      while( true ) {
          if( $a > $maxa ) {
              // done; rest of $used isn't in $all
              break;
          }
          if( $u > $maxu ) {
              // rest of $all is unused
              for( ; $a <= $maxa; $a++ ) {
                  $unused[] = $all[$a];
              }
              break;
          }

          if( $prefix_all.$all[$a] > $prefix_used.$used[$u] ) {
              // $used[$u] isn't in $all?
              $u++;
              continue;
          }

          if( $prefix_all.$all[$a] == $prefix_used.$used[$u] ) {
              // $all[$a] is used
              $a++;
              $u++;
              continue;
```

```
            }

            $unused[] = $all[$a];

            $a++;
        }

        return $unused;
    }
?>
```

up
down
0

ds2u at the hotmail dot com ¶
19 years ago

Yes you can get rid of gaps/missing keys by using:

```
<?php
$result = array_values(array_diff($array1,$array2));
?>
```

But to drop the storage of void spaces (actually a line feed) which are irritatingly indexed when reading from files - just use difference:

```
<?php
$array = array ();
$array[0] = "\n";
$result = array_diff($result,$array);
?>
```

dst

+ add a note

- array_map
- array_merge_recursive
- array_merge
- array_multisort
- array_pad
- array_pop
- array_product
- array_push
- array_rand
- array_reduce
- array_replace_recursive
- array_replace
- array_reverse
- array_search
- array_shift
- array_slice
- array_splice
- array_sum
- array_udiff_assoc
- array_udiff_uassoc
- array_udiff
- array_uintersect_assoc
- array_uintersect_uassoc
- array_uintersect
- array_unique
- array_unshift
- array_values
- array_walk_recursive
- array_walk
- array
- arsort
- asort
- compact
- count
- current
- end
- extract
- in_array
- key_exists
- key
- krsort
- ksort
- list
- natcasesort
- natsort
- next
- pos
- prev
- range
- reset
- rsort
- shuffle
- sizeof
- sort
- uasort
- uksort
- usort

- Deprecated

- each