array_intersect

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

array_intersect — Calcula la intersección de matrices

Descripción ¶

array_intersect (matriz `$array` , matriz `...$arrays`): matriz

array_intersect() devuelve una matriz que contiene todos los valores`array` que están presentes en todos los argumentos. Tenga en cuenta que las claves se conservan.

Parámetros ¶

`array`

La matriz con valores maestros para verificar.

`arrays`

Matrices contra las que comparar valores.

Valores devueltos ¶

Devuelve una matriz que contiene todos los valores `array`cuyos valores existen en todos los parámetros.

Registro de cambios ¶

Versión	Descripción
8.0.0	Esta función ahora se puede llamar con un solo parámetro. Anteriormente, se requerían al menos dos parámetros.

Ejemplos ¶

Ejemplo #1 Ejemplo de array_intersect ()

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

El ejemplo anterior generará:

```
Formación
(
    [a] => verde
    [0] => rojo
)
```

Notas ¶

> **Nota** : Dos elementos se consideran iguales si y sólo si `(string) $elem1 === (string) $elem2`. En palabras: cuando la representación de la cadena es la misma.

See Also ¶

- array_intersect_assoc() - Computes the intersection of arrays with additional index check
- array_diff() - Computes the difference of arrays
- array_diff_assoc() - Computes the difference of arrays with additional index check

+ add a note

User Contributed Notes 33 notes

up
down

154

stuart at horuskol dot co dot uk ¶

14 years ago

A clearer example of the key preservation of this function:

```
<?php

$array1 = array(2, 4, 6, 8, 10, 12);
$array2 = array(1, 2, 3, 4, 5, 6);

var_dump(array_intersect($array1, $array2));
var_dump(array_intersect($array2, $array1));

?>
```

yields the following:

```
array(3) {
  [0]=> int(2)
  [1]=> int(4)
  [2]=> int(6)
}

array(3) {
  [1]=> int(2)
  [3]=> int(4)
  [5]=> int(6)
}
```

This makes it important to remember which way round you passed the arrays to the function if these keys are relied on later in the script.

up

down

68

Niels ¶

16 years ago

Here is a array_union($a, $b):

```
<?php
                                    //  $a = 1 2 3 4
```

```
    $union =                          //  $b =   2   4 5 6
        array_merge(
            array_intersect($a, $b),  //         2   4
            array_diff($a, $b),       //       1   3
            array_diff($b, $a)        //               5 6
        );                            //  $u = 1 2 3 4 5 6
?>
```

up
down
19

Shawn Pyle ¶
13 years ago

array_intersect handles duplicate items in arrays differently. If there are duplicates in the first array, all matching duplicates will be returned. If there are duplicates in any of the subsequent arrays they will not be returned.

```
<?php
array_intersect(array(1,2,2),array(1,2,3)); //=> array(1,2,2)
array_intersect(array(1,2,3),array(1,2,2)); //=> array(1,2)
?>
```

up
down
23

sapenov at gmail dot com ¶
17 years ago

If you need to supply arbitrary number of arguments
to array_intersect() or other array function,
use following function:

```
$full=call_user_func_array('array_intersect', $any_number_of_arrays_here);
```

up
down
14

blu at dotgeek dot org ¶
18 years ago

Note that array_intersect and array_unique doesnt work well with multidimensional arrays.
If you have, for example,

```
<?php
```

```
$orders_today[0] = array('John Doe', 'PHP Book');
$orders_today[1] = array('Jack Smith', 'Coke');

$orders_yesterday[0] = array('Miranda Jones', 'Digital Watch');
$orders_yesterday[1] = array('John Doe', 'PHP Book');
$orders_yesterday[2] = array('Z? da Silva', 'BMW Car');

?>
```

and wants to know if the same person bought the same thing today and yesterday and use array_intersect($orders_today, $orders_yesterday) you'll get as result:

```
<?php

Array
(
    [0] => Array
        (
            [0] => John Doe
            [1] => PHP Book
        )

    [1] => Array
        (
            [0] => Jack Smith
            [1] => Coke
        )

)

?>
```

but we can get around that by serializing the inner arrays:

```
<?php

$orders_today[0] = serialize(array('John Doe', 'PHP Book'));
$orders_today[1] = serialize(array('Jack Smith', 'Coke'));

$orders_yesterday[0] = serialize(array('Miranda Jones', 'Digital Watch'));
```

```
$orders_yesterday[1] = serialize(array('John Doe', 'PHP Book'));
$orders_yesterday[2] = serialize(array('Z? da Silva', 'Uncle Tungsten'));

?>
```

so that array_map("unserialize", array_intersect($orders_today, $orders_yesterday)) will return:

```
<?php

Array
(
    [0] => Array
        (
            [0] => John Doe
            [1] => PHP Book
        )

)

?>
```

showing us who bought the same thing today and yesterday =)

[]s

up
down
8

yuval at visualdomains dot com ¶

7 years ago

Using isset to achieve this, is many times faster:

```
<?php

$m = range(1,1000000);
  $s = [2,4,6,8,10];

// Use array_intersect to return all $m values that are also in $s
  $tstart = microtime(true);
  print_r (array_intersect($m,$s));
```

```
  $tend = microtime(true);
  $time = $tend - $tstart;
  echo "Took $time";

// Use array_flip and isset to return all $m values that are also in $s
  $tstart = microtime(true);
  $f = array_flip($s);
/* $f now looks like this:
(
    [2] => 0
    [4] => 1
    [6] => 2
    [8] => 3
    [10] => 4
)
*/
// $u will hold the intersected values
  $u = [];
  foreach ($m as $v) {
    if (isset($f[$v])) $u[] = $v;
  }
  print_r ($u);
   $tend = microtime(true);
  $time = $tend - $tstart;
  echo "Took $time";
?>

Results:

Array
(
    [1] => 2
    [3] => 4
    [5] => 6
    [7] => 8
    [9] => 10
)
Took 4.7170009613037
(
```

```
    [0] => 2
    [1] => 4
    [2] => 6
    [3] => 8
    [4] => 10
)
Took 0.056024074554443

array_intersect: 4.717
array_flip+isset: 0.056
```

up
down
6

dml at nm dot ru ¶

11 years ago

The built-in function returns wrong result when input arrays have duplicate values.
Here is a code that works correctly:

```
<?php
function array_intersect_fixed($array1, $array2) {
    $result = array();
    foreach ($array1 as $val) {
      if (($key = array_search($val, $array2, TRUE))!==false) {
         $result[] = $val;
         unset($array2[$key]);
      }
    }
    return $result;
}
?>
```

up
down
2

matang dot dave at gmail dot com ¶

7 years ago

Take care of value types while using array_intersect function as there is no option for strict type check as in in_array function.

```
$array1 = array(true,2);
$array2 = array(1, 2, 3, 4, 5, 6);
```

```
var_dump(array_intersect($array1, $array2));

result is :
array(2) {
        [0] =>  bool(true)
        [1]  =>  int(2)
}
```

up
down
1

info at iridsystem dot it ¶
7 years ago

This function is able to sort an array based on another array that contains the order of occurrence. The values that are not present will be transferred into the end of the resultant.
Questa funzione permette di ordinare i valori di un array ($tosort) basandosi sui valori contenuti in un secondo array ($base), i valori non trovati verranno posizionati alla fine dell'ordinamento senza un'ordine specifico.

```
<?
$base= array('one', 'two', 'three');

$tosort= array('a'=>'two', 'b'=>'three', 'c'=>'five', 'd'=>'one', 'and'=>'four', 'f'=>'one');

uasort($tosort, function($key1, $key2) use ($base) {
  $a1=array_search($key1, $base);
  $a2=array_search($key2, $base);

  if ( $a1===false && $a2===false ) { return 0; }
  else if ( $a1===false && $a2 !== false) { return 1; }
  else if ( $a1!==false && $a2 === false) {return -1;}

  if( $a1 > $a2 ) { return 1; }
  else if ( $a1 < $a2 ) { return -1; }
  else if ( $a1 == $a2 ) { return 0; }
});
var_dump($tosort);
/*
the resulting of $tosort
array
```

```
  'd' => string 'one' (length=3)
  'f' => string 'one' (length=3)
  'a' => string 'two' (length=3)
  'b' => string 'three' (length=3)
  'c' => string 'five' (length=6)
  'e' => string 'four' (length=7)
*/
Gabriel
?>
```

up
down
1

Esfandiar -- e.bandari at gmail dot com ¶

14 years ago

Regarding array union: Here is a faster version array_union($a, $b)

But it is not needed! See below.

```
<?php
                                    //  $a = 1 2 3 4
    $union =                        //  $b =   2   4 5 6
        array_merge(
            $a,
            array_diff($b, $a)      //               5 6
        );                          //  $u = 1 2 3 4 5 6
?>
```

You get the same result with $a + $b.

N.B. for associative array the results of $a+$b and $b+$a are different, I think array_diff_key is used.

Cheers, E

up
down
1

theking at king dot ma ¶

6 months ago

I use array_intersect for a quick check of $_GET parameters;

```
<?php declare(strict_types=1)

$_params = ['cust_id','prod_id'];
$_params_check = array_intersect(array_keys($_GET),$_params);
if(count($_params_check) !== count($_params)) {
    header("HTTP/1.1 400 Bad Request");
    die();
}
?>
```

the file will die with a 400 error if cust_id or prod_id or both are missing from $_GET. But i could be used on $_POST and $_REQUEST as well obviously.

up
down
1

Anonymous ¶

1 year ago

```
array_intersect             use Value, not callback
array_uintersect            use Value, callback receives Value
array_intersect_key         use Key, not callback
array_intersect_ukey        use Key, callback receives Key
array_intersect_assoc       use Both, not callback
array_intersect_uassoc      use Both, callback receives Key ONLY
array_uintersect_assoc      use Both, callback receives Value ONLY
array_uintersect_uassoc     use Both, One callback receives the Key, the other receives the Value.
```

up
down
1

Yohann ¶

12 years ago

I used array_intersect in order to sort an array arbitrarly:

```
<?php
$a = array('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'height', 'nine', 'ten');
$b = array('four', 'one', 'height', 'five')
var_dump(array_intersect($a, $b);
?>
```

will output:

```
0 => 'one'
1 => 'four'
2 => 'five'
3 => 'height'
```

i hope this can help...

up
down
0

Oto Brglez ¶

13 years ago

If you wish to create intersection with arrays that are empty. Than the result of intersection is empty array.

If you wish to change this. I sugest that you do this.
It simply "ignores" empty arrays. Before loop use 1st array.

```
<?php

$a = array();
$a[] = 1;
$a[] = 2;
$a[] = 3;

$b = array();
$b[] = 4;
$b[] = 5;
$b[] = 1;

$c = array();
$c[] = 1;
$c[] = 5;
$d = array();

$kb=array('a','b','c','d');

$out = $a;
foreach($kb as $k){
    if(!empty(${$k})) $out = array_intersect($out,${$k});
```

```
};
print_r($out);
// The result is array

// The result is empty array
print_r(array_intersect($a,$b,$c,$d));

?>
```

up
down
0

Malte ¶
14 years ago

Extending the posting by Terry from 07-Feb-2006 04:42:

If you want to use this function with arrays which have sometimes the same value several times, it won't be checked if they're existing in the second array as much as in the first.
So I delete the value in the second array, if it's found there:

```
<?php
$firstarray = array(1, 1, 2, 3, 4, 1);
$secondarray = array(4, 1, 6, 5, 4, 1);

//array_intersect($firstarray, $secondarray): 1, 1, 1, 4

foreach ($firstarray as $key=>$value){
    if (!in_array($value,$secondarray)){
        unset($firstarray[$key]);
    }else{
        unset($secondarray[array_search($value,$secondarray)]);
    }
}

//$firstarray: 1, 1, 4

?>
```

up
down
0

nthitz at gmail dot com ¶

16 years ago

I did some trials and if you know the approximate size of the arrays then it would seem to be a lot faster to do this <?php array_intersect($smallerArray, $largerArray); ?> Where $smallerArray is the array with lesser items. I only tested this with long strings but I would imagine that it is somewhat universal.

up
down
0

t dot wiltzius at insightbb dot com ¶

18 years ago

I needed to compare an array with associative keys to an array that contained some of the keys to the associative array. Basically, I just wanted to return only a few of the entries in the original array, and the keys to the entries I wanted were stored in another array. This is pretty straightforward (although complicated to explain), but I couldn't find a good function for comparing values to keys. So I wrote this relatively straightforward one:

```
<?php

function key_values_intersect($values,$keys) {
   foreach($keys AS $key) {
      $key_val_int[$key] = $values[$key];
      }
   return $key_val_int;
   }

$big = array("first"=>2,"second"=>7,"third"=>3,"fourth"=>5);
$subset = array("first","third");

print_r(key_values_intersect($big,$subset));

?>
```

This will return:

```
Array ( [first] => 2 [third] => 3 )
```

up
down
0

anbolb at boltblue dot com ¶

18 years ago

This is also handy for testing an array for one of a series of acceptable elements. As a simple example, if you're expecting the query string to contain one of, say, user_id, order_id or item_id, to find out which one it is you could do this:

```
<?php
    $valid_ids = array ('user_id', 'item_id', 'order_id');
    if ($id = current (array_intersect ($valid_ids, array_keys ($_GET))))
    {
        // do some stuff with it
    }
    else
        // error - invalid id passed, or none at all
?>
```

...which could be useful for constructing an SQL query, or some other situation where testing for them one by one might be too clumsy.

up
down
-1

terry(-at-)shuttleworths(-dot-)net ¶

16 years ago

I couldn't get array_intersect to work with two arrays of identical objects, so I just did this:

```
foreach ($firstarray as $key=>$value){
    if (!in_array($value,$secondarray)){
        unset($firstarray[$key]);
    }
}
```

This leaves $firstarray as the intersection.

Seems to work fine & reasonably quickly.

up
down
-1

tom p ¶

17 years ago

If you store a string of keys in a database field and want to match them to a static array of values, this is a quick way to do it without loops:

```
<?
```

```
$vals = array("Blue","Green","Pink","Yellow");
$db_field = "0,2,3";

echo implode(", ", array_flip(array_intersect(array_flip($vals), explode(",", $db_field))));

// will output "Blue, Pink, Yellow"

?>
```

up
down
-2

caffinated ¶
10 years ago

If you're looking for a relatively easy way to strictly intersect keys and values recursively without array key reordering, here's a simple recursive function:

```
<?php
function array_intersect_recursive($array1, $array2)
{
  foreach($array1 as $key => $value)
  {
    if (!isset($array2[$key]))
    {
      unset($array1[$key]);
    }
    else
    {
      if (is_array($array1[$key]))
      {
        $array1[$key] = array_intersect_recursive($array1[$key], $array2[$key]);
      }
      elseif ($array2[$key] !== $value)
      {
        unset($array1[$key]);
      }
    }
  }
  return $array1;
```

```
}
?>
```

up
down
-1

ben at kazez dot com ¶

18 years ago

To check whether an array $a is a subset of array $b, do the following:

```
<?php
if(array_unique($b + $a) === $b)
//...
?>
```

Actually, PHP ought to have a function that does this for you. But the above example works.

up
down
-2

david en audiogalaxy punto com ¶

hace 21 años

Note that array_intersect() considers the type of the array elements when it compares them.

If array_intersect() doesn't appear to be working, check your inputs using var_dump() to make sure you're not trying to intersect an array of integers with an array of strings.

arriba
abajo
-3

gary ¶

hace 13 años

i wrote this one to get over the problem i found in getting strings intersected instead of arrays as there is no function in php.

```
<?php
function matched_main_numbers($string, $string2)
{
$string = "04 16 17 20 29";
$arr1 = explode(" ", $string);

$string2 = "45 34 04 29 16";
$arr2 = explode(" ", $string2);
```

```
$array = array_intersect($arr1, $arr2);
$comma_separated = implode($array);

$str = $comma_separated;

$balls = "$comma_separated";
$matched_balls = chunk_split($balls,2," ");
$matched_balls =" $matched_balls";

$number_of_matched_main_balls = strlen($str);
$number_of_matched_main_balls = ($number_of_matched_main_balls/2);
$numbers = "You matched $number_of_matched_main_balls main balls";

return $numbers;

}
?>
```

arriba
abajo
-3

meihao en 126 punto com ¶

Hace 9 años

```
<?php
$a=array(1,2,'3',4);
$b=array('1',2,3);

var_dump($a,$b);

result:
array (size=3)
  0 => int 1
  1 => int 2
  2 => string '3' (length=1)

?>
```

arriba
abajo
-2

Alessandro Ranellucci alex en primafila dot net ¶
hace 19 años

```
array_intersect($array1, $array2);
returns the same as:
array_diff($array1, array_diff($array1, $array2));
```

arriba
abajo
-1

Ehsan.Chavoshi.com ¶
hace 3 años

I wrote this function to recursively replace array keys with array values (flip) and fill values with defined value. it can be used for recursive array intersect functions .

```
<?php
function array_values_to_keys_r($array, $fill_value = null)
{
    $flipped = [];
    foreach ($array as $key => $value) {
        if (is_array($value)) {
            $flipped [$key] = array_values_to_keys_r($value);
        } else {
            $flipped [$value] = $fill_value;
        }
    }
    return $flipped;
}
?>
```

arriba
abajo
-1

zoolyka en gmail punto com ¶
Hace 4 años

If you have to intersect arrays of unique values then using array_intersect_key is about 20 times faster, just have to flip the key value pairs of the arrays, then flip the result again.

```
<?php

$one = range(1, 250000);
$two = range(50000, 150000);
```

```
$start = microtime(true);

$intersection = array_intersect($one, $two);

echo "Did it in ".( microtime(true) - $start )." seconds.\n";

$start = microtime(true);

$intersection = array_flip(array_intersect_key(array_flip($one), array_flip($two)));

echo "Did it in ".( microtime(true) - $start )." seconds.\n";

?>

Did it in 0.89163708686829 seconds.
Did it in 0.038213968276978 seconds.
```

arriba
abajo

-2

carnero ¶

Hace 4 años

```
$x = array('ram@hb.in', 'ram', 'rams@hb.in');
    $y = array('1231231231');
    $result=array_intersect($x,$z);
    $res = array_intersect($y, $z);
```

arriba
abajo

-2

Bloque Mike ¶

Hace 9 años

I bench-marked some uses of array_intersect and can't believe how slow it is. This isn't as elaborate, but handles most cases and is much faster:

```
<?php
/**
examines two arrays and returns the intersected arrays with matching keys (ignores duplicate keys)
*/
function simple_array_intersect($a,$b) {
```

```
    $a_assoc = $a != array_values($a);
    $b_assoc = $b != array_values($b);
    $ak = $a_assoc ? array_keys($a) : $a;
    $bk = $b_assoc ? array_keys($b) : $b;
    $out = array();
    for ($i=0;$i<sizeof($ak);$i++) {
        if (in_array($ak[$i],$bk)) {
            if ($a_assoc) {
                $out[$ak[$i]] = $a[$ak[$i]];
            } else {
                $out[] = $ak[$i];
            }
        }
    }
    return $out;
}

?>
```

You can try this out with this:

```
<?php
// create a large array (simple)
$first = array();
for ($i=500;$i<500000;$i++) {
    $first[] = $i;
}
// create a smaller array (associative)
$second = array();
for ($i=499990;$i<500000;$i++) {
    $second[$i] = rand();
}
echo microtime(true)."\n";
// built-in function
print_r(array_intersect($first,$second));
echo microtime(true)."\n";
// favour simple array as match
print_r(simple_array_intersect($first,$second));
echo microtime(true)."\n";
```

```
// favour associative keys for match
print_r(simple_array_intersect($second,$first));
echo microtime(true)."\n";

?>
```

arriba
abajo
-2
karl en libsyn punto com ¶
hace 13 años

Given a multidimensional array that represents AND/OR relationships (example below), you can use a recursive function with array_intersect() to see if another array matches that set of relationships.

For example: array(array('red'), array('white', 'blue')) represents "red OR (white AND blue)". array('red', array('white', 'blue')) would work, too, BTW.

If I have array('red') and I want to see if it matches the AND/OR array, I use the following function. It returns the matched array, but can just return a boolean if that's all you need:

```
<?php
$needle = array( array( 'red' ), array( 'white', 'blue' ) );
$haystack = array( 'red' );

function findMatchingArray( $needle, $haystack ) {
    foreach( $needle as $element ) {
        $test_element = (array) $element;
        if( count( $test_element ) == count( array_intersect( $test_element, $haystack ) ) ) {
            return $element;
        }

    }
    return false;
}
?>
```

Pretty tough to describe what I needed it to do, but it worked. I don't know if anyone else out there needs something like this, but hope this helps.

arriba
abajo

-5

189780 en gmail punto com ¶

Hace 12 años

Actually array_intersect finds the dublicate values, here is my approach which is 5 times faster than built-in function array_intersect().. Give a try..

```
<?php
function my_array_intersect($a,$b)
{
        for($i=0;$i<sizeof($a);$i++)
        {
                $m[]=$a[$i];
        }
        for($i=0;$i<sizeof($a);$i++)
        {
                $m[]=$b[$i];
        }
        sort($m);
        $get=array();
        for($i=0;$i<sizeof($m);$i++)
        {
                if($m[$i]==$m[$i+1])
                $get[]=$m[$i];
        }
        return $get;
}
?>
```

Barış ÇUHADAR
189780@gmail.com

arriba
abajo

-4

jake ¶

Hace 5 años

Also note that, even without the availability of the strict mode switch, false does not match 0:

```
array_intersect([false], [0]) == [];
```

Even in PHP 7.1

+ añadir una nota

- Funciones de matriz
  - array_change_key_case
  - matriz_trozo
  - matriz_columna
  - array_combinar
  - array_count_values
  - array_diff_asoc
  - array_diff_clave
  - array_diff_uassoc
  - clave_array_diff_u
  - matriz_diferencia
  - array_fill_claves
  - arreglo_relleno
  - array_filtro
  - array_voltear
  - array_intersect_assoc
  - array_intersect_clave
  - array_intersect_uassoc
  - array_intersect_ukey
  - array_intersección
  - array_es_lista
  - array_key_existe
  - array_clave_primero
  - array_key_último
  - array_claves
  - array_mapa
  - array_merge_recursivo
  - matriz_combinar
  - array_multiclasificación
  - array_pad
  - arreglo_pop
  - array_producto
  - array_empujar
  - arreglo_rand
  - array_reducir

- array_replace_recursivo
- matriz_reemplazar
- matriz_reversa
- array_buscar
- matriz_cambio
- array_segmento
- matriz_empalme
- array_suma
- array_udiff_assoc
- array_udiff_uassoc
- array_udiff
- array_uintersect_assoc
- array_uintersect_uassoc
- array_uintersección
- matriz_única
- array_unshift
- matriz_valores
- array_walk_recursivo
- array_caminar
- formación
- ordenar
- un tipo
- compacto
- contar
- Actual
- final
- extracto
- en_matriz
- key_existe
- llave
- ordenar
- sortear
- lista
- natcasesort
- natsort
- Siguiente
- posición
- anterior
- rango

- Reiniciar
- ordenar
- barajar
- tamaño de
- clasificar
- sortear
- ordenar
- usort

- Obsoleto
  - cada