Get next element in foreach loop

Asked 11 years, 8 months ago Modified 8 months ago Viewed 150k times

63

I have a foreach loop and I want to see if there is a next element in the loop so I can compare the current element with the next. How can I do this? I've read about the current and next functions but I can't figure out how to use them.

Thanks in advance

php foreach

Share Edit Follow

asked Feb 23, 2011 at 20:38

chchrist

17.9k 11 45 81

Would it be acceptable to start at element #2 and compare it back to element #1? – Marc B Feb 23, 2011 at 21:01

10 Answers

Sorted by: Highest score (default)

44

A unique approach would be to reverse the array and *then* loop. This will work for non-numerically indexed arrays as well:

```
$items = array(
    'one'   => 'two',
    'two'   => 'two',
    'three' => 'three'
);
$backwards = array_reverse($items);
$last_item = NULL;

foreach ($backwards as $current_item) {
    if ($last_item === $current_item) {
        // they match
    }
    $last_item = $current_item;
}
```

If you are still interested in using the `current` and `next` functions, you could do this:

```
$items = array('two', 'two', 'three');
$length = count($items);
for($i = 0; $i < $length - 1; ++$i) {
    if (current($items) === next($items)) {
        // they match
```

```
    }
}
```

#2 is probably the best solution. Note, `$i < $length - 1;` will stop the loop after comparing the last two items in the array. I put this in the loop to be explicit with the example. You should probably just calculate `$length = count($items) - 1;`

Share Edit Follow

edited Feb 23, 2011 at 21:03

answered Feb 23, 2011 at 20:43

Stephen
18.6k 9 59 98

29

You could probably use while loop instead of foreach:

```
while ($current = current($array) )
{
    $next = next($array);
    if (false !== $next && $next == $current)
    {
        //do something with $current
    }
}
```

Share Edit Follow

edited Apr 12, 2013 at 7:27

Lucas
16.6k 30 106 173

answered Oct 21, 2011 at 13:16

pronskiy
1,339 1 11 8

Just remember, you may have to `reset($array)` prior to this, as the pointer may not be set to the first element. – Jonathan Oct 16, 2015 at 10:39

12

As php.net/foreach points out:

> Unless the array is referenced, foreach operates on a copy of the specified array and not the array itself. foreach has some side effects on the array pointer. Don't rely on the array pointer during or after the foreach without resetting it.

In other words - it's not a very good idea to do what you're asking to do. Perhaps it would be a good idea to talk with someone about why you're trying to do this, see if there's a better solution? Feel free to ask us in ##PHP on irc.freenode.net if you don't have any other resources available.

Share Edit Follow

answered Feb 24, 2011 at 8:05

TML
12.6k 3 36 44

If the indexes are continuous:

12

```
foreach ($arr as $key => $val) {
   if (isset($arr[$key+1])) {
      echo $arr[$key+1]; // next element
   } else {
     // end of array reached
   }
}
```

Share Edit Follow

edited Aug 6, 2018 at 1:06

Tim
2,447 1 23 30

answered Feb 23, 2011 at 20:42

Mārtiņš Briedis
17.1k 5 53 74

4 This is not true, try: array(1 => 'a', 0 => 'b', 100 => 'c'); – Eduardo Romero Oct 10, 2014 at 5:35

16 @EduardoRomero yep, that's why I mentioned: `If indexes ar continious` – Mārtiņš Briedis Oct 10, 2014 at 10:37

You could get the keys/values and index

9

```
<?php
$a = array(
    'key1'=>'value1',
    'key2'=>'value2',
    'key3'=>'value3',
    'key4'=>'value4',
    'key5'=>'value5'
);

$keys = array_keys($a);
foreach(array_keys($keys) as $index ){
    $current_key = current($keys); // or $current_key = $keys[$index];
    $current_value = $a[$current_key]; // or $current_value =
$a[$keys[$index]];

    $next_key = next($keys);
    $next_value = $a[$next_key] ?? null; // for php version >= 7.0

    echo  "{$index}: current = ({$current_key} => {$current_value}); next =
({$next_key} => {$next_value})\n";
}
```

result:

```
0: current = (key1 => value1); next = (key2 => value2)
1: current = (key2 => value2); next = (key3 => value3)
2: current = (key3 => value3); next = (key4 => value4)
3: current = (key4 => value4); next = (key5 => value5)
4: current = (key5 => value5); next = ( => )
```

Share Edit Follow

answered Mar 23, 2018 at 15:17

Andrei Krasutski
4,565 1 27 34

2 Nice one! Works with strings as keys – Lucas Bustamante May 7, 2019 at 16:48

5

if its numerically indexed:

```
foreach ($foo as $key=>$var){

    if($var==$foo[$key+1]){
        echo 'current and next var are the same';
    }
}
```

Share Edit Follow

edited Aug 14, 2012 at 20:39

Rob
400k 70 753 984

answered Feb 23, 2011 at 20:43

user557846

4

The general solution could be a caching iterator. A properly implemented caching iterator works with any Iterator, and saves memory. PHP SPL has a CachingIterator, but it is very odd, and has very limited functionality. However, you can write your own lookahead iterator like this:

```
<?php

class NeighborIterator implements Iterator
{

    protected $oInnerIterator;

    protected $hasPrevious = false;
    protected $previous = null;
    protected $previousKey = null;

    protected $hasCurrent = false;
    protected $current = null;
    protected $currentKey = null;

    protected $hasNext = false;
    protected $next = null;
    protected $nextKey = null;

    public function __construct(Iterator $oInnerIterator)
    {
        $this->oInnerIterator = $oInnerIterator;
    }

    public function current()
    {
        return $this->current;
    }

    public function key()
    {
        return $this->currentKey;
    }

    public function next()
    {
        if ($this->hasCurrent) {
            $this->hasPrevious = true;
```

```
            $this->previous = $this->current;
            $this->previousKey = $this->currentKey;
            $this->hasCurrent = $this->hasNext;
            $this->current = $this->next;
            $this->currentKey = $this->nextKey;
            if ($this->hasNext) {
                $this->oInnerIterator->next();
                $this->hasNext = $this->oInnerIterator->valid();
                if ($this->hasNext) {
                    $this->next = $this->oInnerIterator->current();
                    $this->nextKey = $this->oInnerIterator->key();
                } else {
                    $this->next = null;
                    $this->nextKey = null;
                }
            }
        }
    }

    public function rewind()
    {
        $this->hasPrevious = false;
        $this->previous = null;
        $this->previousKey = null;
        $this->oInnerIterator->rewind();
        $this->hasCurrent = $this->oInnerIterator->valid();
        if ($this->hasCurrent) {
            $this->current = $this->oInnerIterator->current();
            $this->currentKey = $this->oInnerIterator->key();
            $this->oInnerIterator->next();
            $this->hasNext = $this->oInnerIterator->valid();
            if ($this->hasNext) {
                $this->next = $this->oInnerIterator->current();
                $this->nextKey = $this->oInnerIterator->key();
            } else {
                $this->next = null;
                $this->nextKey = null;
            }
        } else {
            $this->current = null;
            $this->currentKey = null;
            $this->hasNext = false;
            $this->next = null;
            $this->nextKey = null;
        }
    }

    public function valid()
    {
        return $this->hasCurrent;
    }

    public function hasNext()
    {
        return $this->hasNext;
    }

    public function getNext()
    {
        return $this->next;
    }

    public function getNextKey()
    {
        return $this->nextKey;
    }
```

```
    public function hasPrevious()
    {
        return $this->hasPrevious;
    }

    public function getPrevious()
    {
        return $this->previous;
    }

    public function getPreviousKey()
    {
        return $this->previousKey;
    }

}


header("Content-type: text/plain; charset=utf-8");
$arr = [
    "a" => "alma",
    "b" => "banan",
    "c" => "cseresznye",
    "d" => "dio",
    "e" => "eper",
];
$oNeighborIterator = new NeighborIterator(new ArrayIterator($arr));
foreach ($oNeighborIterator as $key => $value) {

    // you can get previous and next values:

    if (!$oNeighborIterator->hasPrevious()) {
        echo "{FIRST}\n";
    }
    echo $oNeighborIterator->getPreviousKey() . " => " . $oNeighborIterator-
>getPrevious() . " -----> ";
    echo "[ " . $key . " => " . $value . " ] -----> ";
    echo $oNeighborIterator->getNextKey() . " => " . $oNeighborIterator-
>getNext() . "\n";
    if (!$oNeighborIterator->hasNext()) {
        echo "{LAST}\n";
    }
}
```

Share Edit Follow

answered Mar 14, 2016 at 19:50

Dávid Horváth
3,914 1 21 33

2

A foreach loop in php will iterate over a copy of the original array, making `next()` and `prev()` functions useless. If you have an associative array and need to fetch the next item, you could iterate over the array keys instead:

```
foreach (array_keys($items) as $index => $key) {
    // first, get current item
    $item = $items[$key];
    // now get next item in array
    $next = $items[array_keys($items)[$index + 1]];
}
```

Since the resulting array of keys has a continuous index itself, you can use that instead to access the original array.

Be aware that `$next` will be `null` for the last iteration, since there is no next item after the last. Accessing non existent array keys will throw a php notice. To avoid that, either:

1. Check for the last iteration before assigning values to `$next`
2. Check if the key with `index + 1` exists with `array_key_exists()`

Using method 2 the complete foreach could look like this:

```
foreach (array_keys($items) as $index => $key) {
    // first, get current item
    $item = $items[$key];
    // now get next item in array
    $next = null;
    if (array_key_exists($index + 1, array_keys($items))) {
        $next = $items[array_keys($items)[$index + 1]];
    }
}
```

Share Edit Follow

edited Feb 14, 2016 at 16:39

answered Feb 14, 2016 at 16:28

Liquinaut
3,621 1 20 17

2 Using array_keys on each iteration? This must be the slowest solution. – Semra Jul 22, 2017 at 11:04

2

You could get the keys of the array before the foreach, then use a counter to check the next element, something like:

```
//$arr is the array you wish to cycle through
$keys = array_keys($arr);
$num_keys = count($keys);
$i = 1;
foreach ($arr as $a)
{
    if ($i < $num_keys && $arr[$keys[$i]] == $a)
    {
        // we have a match
    }
    $i++;
}
```

This will work for both simple arrays, such as `array(1,2,3)`, and keyed arrays such as `array('first'=>1, 'second'=>2, 'thrid'=>3)`.

Share Edit Follow

edited Feb 24, 2011 at 9:36

answered Feb 23, 2011 at 21:10

eclipse31
364 4 15

IMO, smartest way of doing it. – Velojet Mar 21, 2018 at 2:45

0

```php
$next_data = $data;

$prev_key = null;
$prev_value = null;

foreach($data as $key => $value)
{
    array_shift($next_data);

    $next_key = key($next_data);
    $next_value = $next_data[$next_key] ?? null;

    // Do something here...

    $prev_key = $key;
    $prev_value = $value;
}
```

Share Edit Follow

answered Feb 13 at 15:58

SirCumz

171 1 2 14