

Arrays

Arr of Derafu Support

Anonymous · 08/10/2026

Arrays

Arr has the transformations of arrays that come up again and again: flat and nested forms, casting, trees, tables and subsets.

Method	What it does
<code>Arr::dot(\$array, \$prefix = '')</code>	Flattens an array into keys with dots.
<code>Arr::nested(\$array)</code>	The opposite: keys with dots into an array of arrays.
<code>Arr::cast(\$array, \$emptyValue = '')</code>	Casts the values of an array, recursively.
<code>Arr::ensureArrayAtPath(&\$array, \$path)</code>	Makes sure that the value at a path is a list.
<code>Arr::ensureIdInElements(\$data, \$idField)</code>	Puts the key of each element in it, as its id.
<code>Arr::subsets(\$array, \$minLength = 1)</code>	All the subsets of an array.
<code>Arr::toTree(\$items, \$parentField, \$childrenField, \$parentId = null)</code>	A flat array with references to the parent into a tree.
<code>Arr::treeToList(\$tree, \$nameField, \$childrenField)</code>	A tree into a flat list with levels.
<code>Arr::groupToTable(\$array, \$keys = null)</code>	<code>['k' => [1, 2]]</code> into rows.
<code>Arr::tableToAssociative(\$table)</code>	A table of two columns into an associative array.
<code>Arr::toXml(\$array, \$rootElement = 'root')</code>	An array into XML.

Dot notation

```
use Derafu\Support\Arr;

Arr::dot(['a' => ['b' => 1, 'c' => [2, 3]]]);
// ['a.b' => 1, 'a.c.0' => 2, 'a.c.1' => 3]
```

```
Arr::nested(['a.b' => 1, 'a.c' => 2, 'd' => 3]);
// ['a' => ['b' => 1, 'c' => 2], 'd' => 3]
```

`dot()` takes a `$prefix` that is put before every key (`Arr::dot(['b' => 1], 'a')` is `['a.b' => 1]`).

Casting

`cast()` goes through the array, and in all its levels it trims the texts, turns the numeric texts into `int` or `float`, and replaces the empty texts with `$emptyValue` ('' by default, which leaves them as they are).

```
Arr::cast(['a' => ' x ', 'b' => '', 'c' => ['d' => '5']], null);
// ['a' => 'x', 'b' => null, 'c' => ['d' => 5]]
```

It does not turn `true` or `null` written as texts into values: for that, see [Caster](#).

Trees

`toTree()` uses the **keys** of the array as the identifiers of the items, and the field `$parentField` of each item has the **key** of its parent. The items of the root are the ones whose parent is `$parentId` (`null` by default). The field of the parent is removed from each item, and the children go in `$childrenField`:

```
$items = [
    1 => ['parent_id' => null, 'name' => 'root'],
    2 => ['parent_id' => 1, 'name' => 'a'],
    3 => ['parent_id' => 2, 'name' => 'b'],
    4 => ['parent_id' => 1, 'name' => 'c'],
];

$tree = Arr::toTree($items, 'parent_id', 'children');
// [1 => ['name' => 'root', 'children' => [
//     2 => ['name' => 'a', 'children' => [3 => ['name' => 'b', 'children' => []]]],
//     4 => ['name' => 'c', 'children' => []],
// ]]]
```

If the items come from a database as a list, key them by their id first (`array_column($rows, null, 'id')`). An item without the field of the parent is skipped.

`treeToList()` goes the other way, in depth order. Each element has its `name` (from `$nameField`) and its `level` (0 for the root), under the same key that it had:

```
Arr::treeToList($tree, 'name', 'children');
// [1 => ['name' => 'root', 'level' => 0], 2 => ['name' => 'a', 'level' => 1],
```

```
// 3 => ['name' => 'b', 'level' => 2], 4 => ['name' => 'c', 'level' => 1]]
```

Tables

```
Arr::groupToTable(['k1' => [1, 2], 'k2' => [3, 4]]);
// [['k1' => 1, 'k2' => 3], ['k1' => 2, 'k2' => 4]]

Arr::tableToAssociative(['a', 1], ['b', 2]);
// ['a' => 1, 'b' => 2]
```

`tableToAssociative()` needs exactly two columns in each row; a row that does not have them is an `InvalidArgumentException` that says which one.

Other helpers

```
Arr::subsets([1, 2, 3]);
// [[1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]

Arr::subsets([1, 2, 3], 2);
// [[1, 2], [1, 3], [2, 3], [1, 2, 3]]

Arr::ensureIdInElements(['u1' => ['n' => 1], 'u2' => ['n' => 2]], 'id');
// ['u1' => ['id' => 'u1', 'n' => 1], 'u2' => ['id' => 'u2', 'n' => 2]]

Arr::toXml(['a' => 1, 'b' => ['c' => 'x&y']], 'r');
// <?xml version="1.0"?><r><a>1</a><b><c>x&y</c></b></r>
```

`subsets()` gives 2^n subsets: do not use it with more than a few elements. `ensureArrayAtPath()` changes the array that it is given (it is passed by reference): if the value at the path is a single associative array, it becomes a list with that array as its only element.

Last updated on 08/10/2026