Duplicates

Find the items of a list that repeat the values of another one, comparing by paths at any level

Anonymous · 05/11/2026

ArrayDuplicates

`ArrayDuplicates` finds the items of a list that repeat the values of an earlier item. What identifies an item are the values at the paths that are given, which can be at any level; without paths, the whole item is compared.

```
use Derafu\Array\ArrayDuplicates;

$documents = [
    ['TpoDoc' => 33, 'NroDoc' => 21],
    ['TpoDoc' => 33, 'NroDoc' => 22],
    ['TpoDoc' => 61, 'NroDoc' => 21],
    ['TpoDoc' => 33, 'NroDoc' => 21],
];

ArrayDuplicates::find($documents, ['TpoDoc', 'NroDoc']);
// [['index' => 3, 'first' => 0, 'value' => [33, 21]]]
```

Method	What it does
`find($items, $paths = [], $normalize = null)`	Every item that repeats an earlier one.
`first(...)`	The first one, or `null` if there is none.
`assertUnique(...)`	Throws `DuplicateValueException` for the first one. Returns nothing.

What Is Reported

Each repeated item is reported with three entries:

Entry	Content
`index`	The key of the repeated item in the list, as it is in the list.
`first`	The key of the first item that has those values.
`value`	The values read from the repeated item, in the order of the paths, as they are (before normalizing them).

The items are reported in the order of the list. An item that appears three times is reported twice, each one pointing to the first.

Paths

A path is a string, an integer or a list of them, and it is read with `ArrayPath::get()`: a key, dot notation (`'Encabezado.IdDoc.Folio'`) or the levels as a list (`['Encabezado', 'IdDoc', 'Folio']`). Paths of different levels can be mixed in the same call.

```
ArrayDuplicates::find($items, ['a', 'b.c']);
```

A path that is not in an item gives `null`, so two items that do not have it have the same value for it. An item that is an object is read through its public properties. An item that is neither an array nor an object can not be read by paths and is an `ArrayException`, as is a path that is not valid.

Without paths the whole item is compared, whatever it is: an array, an object or a scalar.

Comparison

Two items are the same when their values have the same type and the same value: `'1'` and `1` are different. The values must be serializable; one that is not (a closure, for instance) is an `ArrayException` that tells which item has it.

To compare in another way, pass `$normalize`, a `callable(mixed): mixed` that is applied to each value that is read before comparing it:

```
$normalize = static fn (mixed $value): string => (string) (int) trim((string) $value);

ArrayDuplicates::find([['n' => 21], ['n' => '0021'], ['n' => ' 21 ']], ['n'],
$normalize);
// [
//     ['index' => 1, 'first' => 0, 'value' => ['0021']],
//     ['index' => 2, 'first' => 0, 'value' => [' 21 ']],
// ]
```

The items are compared in one pass, so the time grows with the size of the list.

Requiring Unique Items

`assertUnique()` is for who only needs to know that there are no repeated items. Like `assertOnlyKeys()`, it checks a property of the whole data and throws a specific exception:

```
use Derafu\Array\Exception\DuplicateValueException;
```

```
try {
    ArrayDuplicates::assertUnique($documents, ['TpoDoc', 'NroDoc']);
} catch (DuplicateValueException $exception) {
    echo $exception->getMessage();
    // The item with key 3 repeats the item with key 0 (TpoDoc = 33, NroDoc = 21).

    $exception->getIndex();  // 3
    $exception->getFirst();  // 0
    $exception->getValues(); // [33, 21]
}
```

`DuplicateValueException` extends `ArrayException` and carries the key of the repeated item, the key of the first one and the values, so whoever catches it can say it in its own words. The message names the items by their key in the list, which can be a number or a text, and shows each path with the value that is repeated. Without paths it is `The item with key {index} is the same as the item with key {first}.` See Translations for its messages.

Last updated on 05/11/2026