TYPO3.Flow - Feature # 32985

Status:	New	**Priority:**	Should have
Author:	Sebastian Kurfuerst	**Category:**	Utility
Created:	2012-01-05	**Assigned To:**	
Updated:	2012-05-30	**Due date:**	
PHP Version:			
Has patch:	No		
Complexity:			
Subject:	Implement Processing Rules when merging numerically-indexed arrays		

Description

This is an improvement to the Utility\Arrays::arrayMergeRecursiveOverrule function, leading to enormous flexibility gains for YAML Configuration Handling.

Description of the problem and possible solution:

```
* Sometimes, you need to merge numerically indexed PHP arrays; and as this function does
* the merging by key, this can lead to undesired results, as the following example demonstrates:
*
* $array1 = array('a1', 'b1', 'c1', 'd1');
* $array2 = array('a2', 'b2', 'c2');
* -> the resulting array is now array('a2', 'b2', 'c2', 'd1'), which is usually
*    not the desired result. Often, you want to append, prepend or replace the collection.
*
* To define the behavior with indexed collections, a *Processing Rule* can be specified
* in the second array using the special *__processingRule* Array Key. The following
* Processing Rules are supported:
*
* APPEND
* ------
* The most common processing rule. Appends $array2 to $array1, renumbering the
* numerical indices in $array2 and (if necessary) overriding the associative indices in $array1.
*
* Example:
* $array1 = array('a1', 'b1', 'c1', 'd1');
* $array2 = array('__processingRule' => 'APPEND', 'a2', 'b2', 'c2');
* --> Result: array('a1', 'b1', 'c1', 'd1', 'a2', 'b2', 'c2');
*
* PREPEND
* -------
* Prepends $array2 before $array1, renumbering the numerical indices in
* $array1 and (if necessary) overriding the associative indices in $array2.
*
* Example:
* $array1 = array('a1', 'b1', 'c1', 'd1');
* $array2 = array('__processingRule' => 'PREPEND', 'a2', 'b2', 'c2');
* --> Result: array('a2', 'b2', 'c2', 'a1', 'b1', 'c1', 'd1');
*
* REPLACE
* -------
* Completely replaces $array1 with $array2.
```

```
     *
     * Example:
     * $array1 = array('a1', 'b1', 'c1', 'd1');
     * $array2 = array('__processingRule' => 'REPLACE', 'a2', 'b2', 'c2');
     * --> Result: array('a2', 'b2', 'c2');
```

With this addition, the following becomes possible in YAML configuration files:

```
// per-package Settings.yaml

  Foo:
    Bar: [t1, t2]

// global Settings.yaml

  Foo:
    Bar:
      __processingRule: PREPEND
      0: o2
      1: o3

// Output:
array('Foo' => array('Bar' => array('o2', 'o3', 't1', 't2')));
```

As soon as the Symfony YAML parser supports custom callbacks for **YAML Tags**, the processing rules could be written as:

```
  Foo:
    Bar: !PREPEND
      0: o2
      1: o3
```

... which is again even nicer :)

History

#1 - 2012-01-05 08:27 - Gerrit Code Review

- Status changed from Accepted to Under Review

Patch set 1 for branch **master** has been pushed to the review server.
It is available at http://review.typo3.org/7677

#2 - 2012-01-05 10:02 - Bastian Waidelich

I'm not too happy with this, to be honest, for a few reasons (just IMO of course):

- arrayMergeRecursiveOverrule() is called **very** often. So it should be as simple and fast as possible and it shouldn't contain some higher level processing logic.
- I think it's not practical to have processing instructions in the array as that could have unpredictable side effects. For YAML tags that's different because the Settings.yaml is some kind of DSL anyways - but it shouldn't be done for every array. But even for "YAML arrays" it would be good if we didn't need this too often as it makes it harder to read and comprehend.

I'm not sure for what case exactly you need this but it might be possible to achieve this by restructuring the configuration (so that it has keys that can be overwritten) or to put the processing logic into the consuming entity (ConfigurationManager / xyzParser)

#3 - 2012-01-05 10:26 - Karsten Dambekalns

Bastian Waidelich wrote:

> *I'm not too happy with this, to be honest, for a few reasons (just IMO of course):*

I agree, and REPLACE seems pretty useless from a PHP perspective. It only makes sense in a YAML context.

> *I'm not sure for what case exactly you need this but it might be possible to achieve this by restructuring the configuration (so that it has keys that can be overwritten) or to put the processing logic into the consuming entity (ConfigurationManager / xyzParser)*

That sounds like the better solution to me.

#4 - 2012-01-29 18:34 - Sebastian Kurfuerst

- Status changed from Under Review to New
- Assigned To deleted (Sebastian Kurfuerst)

de-assigning because we do not have any consensus if we need that feature.

#5 - 2012-05-21 16:15 - Karsten Dambekalns

- Target version deleted (1.1)

#6 - 2012-05-30 09:34 - Christian Müller

We need a consensus here, for example the current merging strategy leads to the following:

FLOW3 Package ignoreTags:
ignoredTags: ['api', 'package', 'subpackage', 'license', 'copyright', 'author', 'const', 'see', 'todo', 'scope', 'fixme', 'test', 'expectedException', 'depends', 'dataProvider', 'group', 'codeCoverageIgnore']

Now you want to add something in your package and do:
ignoredTags: ['foo', 'bar']

The result will be:

array('foo', 'bar', 'subpackage', 'license', 'copyright', 'author', 'const', 'see', 'todo', 'scope', 'fixme', 'test', 'expectedException', 'depends', 'dataProvider', 'group', 'codeCoverageIgnore')

as the merge is key based. This leads to strange errors about tags, but I guess there can be more problematic results in other settings... Only way atm is to copy the full list and add to that, which of course is not very good to update...

#7 - 2012-05-30 09:35 - Christian Müller

Additionally there is no way to empty an array (unless you override it with another type, so you set it to an empty string)