TYPO3.Fluid - Feature # 6289

Status:	Resolved	**Priority:**	Should have
Author:	Franz Koch	**Category:**	ViewHelpers
Created:	2010-01-30	**Assigned To:**	Sebastian Kurfuerst
Updated:	2010-10-20	**Due date:**	
Has patch:			
Subject:	{settings} not automatically available in partials		

Description

Hi, I just noticed that the {settings} array is not automatically available in partials, which it should in my eyes.
This is the v4 branch, not sure if it's also the case in v5.

Associated revisions

Revision 5beb4177 - 2010-09-08 11:05 - Sebastian Kurfuerst

[+FEATURE] Fluid (Core): {settings} is available in Partials

Now, the {settings} are automatically available in partials and sections.
Before, they had to be passed explicitely, i.e. by calling
<f:render section="..." arguments="{settings: settings}" />.

If somebody defined his own "settings"-argument, this still takes precedence:
<f:render section="..." arguments="{settings: some.different.settings}" />
This means that this change is completely backwards compatible.

NOTE: The settings are NOT merged together, like it has been suggested in #6289,
as this will lead to un-obvious behavior for the user.

The test case for this change can be found in I414a82f7beb0e13a615ed39acde93d3c3546ccc1
so make sure to apply these two patches together.

Change-Id: I14c6dd4d9db13a98f1873a79bb43fbf299d2c835
Resolves: #6289

History

#1 - 2010-03-15 16:46 - Sebastian Kurfuerst

- Status changed from New to Accepted
- Assigned To set to Sebastian Kurfuerst
- Priority changed from Should have to Must have

#2 - 2010-06-18 10:04 - Sebastian Kurfuerst

- Status changed from Accepted to New
- Assigned To deleted (Sebastian Kurfuerst)
- Priority changed from Must have to Should have

we still need to discuss how to change this in a backwards-compatible way.

#3 - 2010-07-13 11:34 - Sebastian Kurfuerst

- Status changed from New to Accepted

- Assigned To set to Sebastian Kurfuerst

Felix wants to work on this.

#4 - 2010-07-28 09:56 - Steffen Ritter

- File fluidIssue6289.diff added

attached one-liner solves this issue...

#5 - 2010-07-28 10:00 - Felix Oertel

Great, thanks. Should be

$arguments = t3lib_div::array_merge_recursive_overrule(array('settings' => $this->templateVariableContainer->get('settings')), $arguments);

I think. Settings given in the method's arguments should allway overrule.

#6 - 2010-07-28 10:03 - Steffen Ritter

Felix Oertel wrote:

> *Great, thanks. Should be*
>
> *$arguments = t3lib_div::array_merge_recursive_overrule(array('settings' => $this->templateVariableContainer->get('settings')), $arguments);*
>
> *I think. Settings given in the method's arguments should allway overrule.*

yeah right :) fine with this, too...

#7 - 2010-07-28 10:34 - Bastian Waidelich

- Category set to ViewHelpers

- Branch changed from v4 to v4 + v5

Hi all,

I don't like to play the grinch, but I'm not sure if we really want this feature..
In the mailing list I wrote

> partials are meant to be reusable Fluid snippets. You should be able to even use them from other extensions if you want to.
> So they should not rely on some n nongeneric settings array but rather on some explicit options.
> E.g. if you want to render your form text fields in a similar manner, you could have a partial like:
>
> <dt{f:if(condition: required, then: ' class="required")}>

```
    {label}:
</dt>
<dd>
    <f:form.textbox property="{propertyName}" />
</dd>
```

And now you could render it

```
<f:render partial="form.myTextField" agruments="{label: 'First Name', propertyName: 'firstName', required: 'true'}" />
```

-> More to write, but less intransparent and error-prone.

Don't you agree?

#8 - 2010-07-28 11:46 - Franz Koch

Hey Bastian,
I don't think that providing the settings in partials would conflict with this concept - especially when it comes to widgets at some time.
In your example, you're using a partial that of course should/has to be configured by passed arguments/options.
But imagine a partial where you allow to configure css class names, colors or a paginagtion related configuration that should be globally changeable via TS. You could use these special layout/design related options from the settings array, easy to adopt for various extensions via TS then, without having to mess around with their default templates or replace the default partial with your customized one etc.
Or what if you like to use a widget in a partial that depends on some configurations options from the settings array?

#9 - 2010-07-28 15:02 - Sebastian Kurfuerst

Hi,

I think as well that it would be a helpful feature. To me, partials are useful in two cases:

1. to encapsulate similar parts across **one** extension
2. to encapsulate similar parts across **multiple** extensions.

While I agree with Bastian that in the second case, depending on settings is ugly and non-intuitive, I think it is quite intuitive in the first case -- and it really helps to remove duplication in a **single** extension.

Bastian, could I somehow convince you? :-)

Greets,
Sebastian

#10 - 2010-08-06 21:26 - Bastian Waidelich

Sebastian Kurfuerst wrote:

> *Bastian, could I somehow convince you? :-)*

gn gn.. not really ;)

compare partials with global functions for a minute.
Of course you could have all method parameters in one array like

```
public function doSomething(array $options)
```

This might even make it easier to extend the function later.
But it's very error-prone as you are not working on some kind of "common contract".

> *To me, partials are useful in two cases: 1. to encapsulate similar parts across **one** extension*
> *2. to encapsulate similar parts across **multiple** extensions.*

I think, even for the first case it makes sense to move your partial configuration into one "section" of your global settings:

```
settings {
  somethingCompletelyDifferent = 123
  defaultColor = red

  somePartial {
    defaultCssClass = foo
    defaultColor < tx_pluginsignature.settings.defaultColor
  }
}
```

and use it like

```
<f:render partial="yourPartial" arguments="{cssClass: settings.somePartial.defaultCssClass, color: settings.somePartial.defaultColor} />
```

or - if you want to reduce the amount of arguments -

```
<f:render partial="yourPartial" arguments="{settings: settings.yourPartial} />
```

And, finally, if you really want to share all plugin settings with the partial, you can still set them as argument.. But I'm not in favor of implicitly merging them..

#11 - 2010-09-08 11:08 - Sebastian Kurfuerst

Hey,

this issue is now in Gerrit, waiting for reviews.

Issue: https://review.typo3.org/64
Test Case: https://review.typo3.org/62

NOTE: The settings are NOT merged together, like it has been suggested in #6289,
as this will lead to un-obvious behavior for the user.

Greets,
Sebastian

#12 - 2010-09-09 10:45 - Sebastian Kurfuerst

- Status changed from Accepted to Resolved

- % Done changed from 0 to 100

Applied in changeset commit:"5beb4177be2efff10cfe396d940e8d59f176ca64".

Files

fluidIssue6289.diff	634 Bytes	2010-07-28	Steffen Ritter