TYPO3.Fluid - Feature #7608

Configurable shorthand/object accessor delimiters

2010-05-03 15:31 - Lienhart Woitok

Status:	New	**Start date:**	2010-05-03
Priority:	Could have	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Core	**Estimated time:**	0.00 hour
Target version:			
Has patch:	Yes		

Description

See Mailinglist Thread "About limitations in Fluid"

For some output formats (like LaTeX) it is not comfortable to use { and } for shorthand syntax as these symbols are used by the target language. Therefore, the possibility to configure the open and close symbols for short hand syntax is necessary.

Attached patch provides this.

Related issues:

Related to TYPO3.Fluid - Feature #43356: Allow Fluid arrays only in ViewHelpe...	**Resolved**	**2012-11-27**
Has duplicate TYPO3.Fluid - Feature #11286: Alternative syntax for fluid obje...	**Closed**	**2010-12-05**

History

#1 - 2010-05-04 00:09 - Thomas Deinhamer

Just my 2 cents, maybe it's nicer to name the variables
$openingShorthandSymbol and $closingShorthandSymbol or
maybe there are even better names for them.

#2 - 2010-05-07 07:21 - Jochen Rau

- Assignee set to Sebastian Kurfuerst

#3 - 2010-06-18 15:20 - Sebastian Kurfuerst

- Project changed from 534 to TYPO3.Fluid

- Assignee deleted (Sebastian Kurfuerst)

#4 - 2010-06-18 15:21 - Sebastian Kurfuerst

- Priority changed from Should have to Could have

#5 - 2010-11-14 12:17 - Bastian Waidelich

- Subject changed from Configurable short hand syntax for fluid to Configurable shorthand/object accessor delimiters

- Category set to Core

- Branch set to v4 + v5

#6 - 2011-05-06 10:30 - Sebastian Kurfuerst

- Has patch set to Yes

#7 - 2012-11-26 16:07 - Karsten Dambekalns

One issue with such an approach is: what happens if you change this, and thus break templates included with other people's packages? The delimiter used should be configured in the template, then you could mix it without worries…

#8 - 2012-11-26 16:11 - Bastian Waidelich

Karsten Dambekalns wrote:

> One issue with such an approach is: what happens if you change this, and thus break templates included with other people's packages? The delimiter used should be configured in the template, then you could mix it without worries…

I just thought the same.. it could be set in initializeView for one action only somehow but that would couple controller & view too tightly together. An alternative solution would be a ViewHelper that initializes the parser itself to parse the child nodes – but still we'd need to be able to configure the parser and therefore need something similar to Lienhards solution..

#9 - 2012-11-26 17:44 - Bastian Waidelich

- File fluid-shorthandsyntax_7608_v2.patch added

Attached an updated (and slightly adjusted) version of Lienharts patch.
What's missing now is a way to modify the parser configuration from the "outside" (we need that anyways) and a way to specify the opening/closing symbols in the template.
I created a basic test view helper for that:

```
class TestViewHelper extends \TYPO3\Fluid\Core\ViewHelper\AbstractViewHelper {

    /**
     * @var \TYPO3\Fluid\View\StandaloneView
     * @Flow\Inject
     */
    protected $view;

    /**
     * @param string $openingShorthandSymbol
     * @param string $closingShorthandSymbol
     * @return string
     */
    public function render($openingShorthandSymbol = '{', $closingShorthandSymbol = '}') {
        $this->view->setTemplateSource($this->renderChildren());
        $this->view->setControllerContext($this->renderingContext->getControllerContext());
        $this->view->assignMultiple($this->templateVariableContainer->getAll());

            // NOTE: This getter doesn't exist yet
        $templateParser = $this->view->getTemplateParser();
        $parserConfiguration = $templateParser->getConfiguration();
        if ($parserConfiguration === NULL) {
            $parserConfiguration = new \TYPO3\Fluid\Core\Parser\Configuration();
        }
        $parserConfiguration->setOpeningShorthandSymbol($openingShorthandSymbol);
        $parserConfiguration->setClosingShorthandSymbol($closingShorthandSymbol);
            // NOTE: ...neither does this setter
        $templateParser->setConfiguration($parserConfiguration);

        return $this->view->render();
    }
}
```

That could be used like

```
<x:test openingShorthandSymbol="###" closingShorthandSymbol="###"><![CDATA[
    ###someObjects -> f:count()###
]]></x:test>
```

(note the required CDATA tags for this)

I don't like it. Maybe something like the {namespace ...} syntax would be nicer, but then this can only work per template/partial..

#10 - 2012-11-28 10:56 - Bastian Waidelich

FYI: After discussing this again with Sebastian we still did not find a nice way to configure this. Because it has to be changeable per template/partial/layout it would probably need to be defined **in the file** or via the upcoming **Views.yaml**.. For now have a look at #43356 which greatly reduces the clashes if you use Fluid in JavaScript, CSS etc.

#11 - 2013-09-23 18:09 - Bastian Waidelich

FYI there is a neat little work around using the **alias** ViewHelper.

So instead of having to use CDATA everywhere to escape curly brackets:

```
<script>
```

```
var options = <![CDATA[{]]>
  foo = "{foo}",
  bar = "{bar}
<![CDATA[}]]>;
function foo() <![CDATA[{]]>

<![CDATA[}]]>
</script>
```

you can also put your content in a alias VH:

```
<f:alias map="{l: '{', r: '}'}">
<script>
var options = {l}
  foo = "{foo}",
  bar = "{bar}
{r};
function foo() {l}

{r}
</script>
</f:alias>
```

#12 - 2013-11-07 11:22 - Wouter Beeftink

I would love to have an alternative delimiter for fluid tags. Bastian's approach is a great workaround for now.
If it were up to me this ticket should be labelled as **should have**.

Files

fluid-shorthandsyntax.patch	9.84 KB	2010-05-03	Lienhart Woitok
fluid-shorthandsyntax_7608_v2.patch	17.5 KB	2012-11-26	Bastian Waidelich