TYPO3.Fluid - Feature # 30555

Status:	New	**Priority:**	Could have
Author:	Thomas Allmer	**Category:**	Core
Created:	2011-10-04	**Assigned To:**	
Updated:	2012-06-29	**Due date:**	
Has patch:	No		
Subject:	Make TagBuilder more extensible		

Description

As a developer for Viewhelpers I would like to Extend available Viewhelper and probably modify some tag attributes based on various options without totally forking the viewhelper.

For that I need the possibility to get and set the value of tag attributes.

These Functions could look like this:

```
/**
 * Sets the value of an attribute in the $attributes-collection
 *
 * @param string $attributeName name of the attribute to be added to the tag
 * @return string
 * @author Thomas Allmer <at@delusionworld.com>
 * @api
 */
public function setAttribute($attributeName, $attributeValue, $escapeSpecialCharacters = TRUE) {
    return $this->addAttribute($attributeName, $attributeValue, $escapeSpecialCharacters);
}

/**
 * Gets the value of an attribute in the $attributes-collection
 *
 * @param string $attributeName name of the attribute to be added to the tag
 * @return string
 * @author Thomas Allmer <at@delusionworld.com>
 * @api
 */
public function getAttribute($attributeName) {
    return $this->attributes[$attributeName] ? $this->attributes[$attributeName] : '';
}

/**
 * Checks if the tag has a certain attribute
 *
 * @param string $attributeName name of the attribute to be added to the tag
 * @return string
 * @author Thomas Allmer <at@delusionworld.com>
 * @api
 */
public function hasAttribute($attributeName) {
    return $this->attributes[$attributeName] ? TRUE : FALSE;
}
```

Related issues:

related to TYPO3.Fluid - Feature # 37460: TagBuilder should allow access to a...	**Resolved**	**2012-05-24**

History

#1 - 2011-10-18 09:55 - Bastian Waidelich

- Subject changed from TagBuilder does not allow to work with tags to Make TagBuilder more extensible

- Category set to Core

- Priority changed from Must have to Could have

I agree that the TagBuilder could be more flexible.
Here some more ideas that came up during a discussion in the FLOW3 Mailing List:

> *Currently, the attributes are "just" an array with attribute names as keys and their values as... values.*
> *In HTML, sometimes it comes to "collections" inside attributes, I think especially about "class" and "style". Multiple class names resp. style directives are allowed inside a class/style attribute.*
> *[...]*
> *My thought now is, that it would be nice to make attributes configurable to hold collections, with a per-attribute definable separator (that would be [space] on "class", and [;] on "style".*

Concerning the idea to make addAttribute() adding the value to the existing attribute - I'm not sure about that one. It would be a bit intransparent and only really make sense for the class & style attribute. Instead I'd suggest to introduce setAttribute() (as suggested) which ands or replaces existing attribute and something like appendAttribute() that extends the value.

#2 - 2011-10-18 13:16 - Thomas Allmer

I would suggest something like this (concept from MooTools)

```
setAttribute('foo', 'bar'); // <tag foo="bar" />
getAttribute
hasAttribute
eraseAttribute OR removeAttribute

setStyle('border', 'none'); // <tag style="border: none;" />
getStyle

addClass('myClass'); // <tag class="myClass" />
removeClass

//advanced Class stuff
hasClass
addClass
removeClass
toggleClass

//convinients function
setStyles
```

getStyles
setAttributes
getAttributes
eraseAttributes or removeAttributes

so what do you think? should I work on something like this?

#3 - 2012-06-29 12:41 - Adrian Föder

since basic functionality is indeed resolved in #37460, I still, of course ;) like Thomas' Mootoolsian ideas, however I would not go so far and support a whole convenient feature set, since e.g. toggleClass and similar maybe don't make much sense in that "static" context.

what I could imagine as a convenient, powerful method is, a method like
[pseudo-code]

```
function appendAttributeValue(string $attribute, string $value, string $assureSeparator = NULL)
 if (! attributeExists) createAndSetIt;
 else {
    if (lastPartOfExistingValue != $assureSeparator) setAttribute($currentValue . $assureSeparator . $value)
    else {
        setAttribute($currentValue . $value);
    }
}
```

this is universally usable for styles, classes and javascript-code and the user just has to care about using the correct @assureSeparator@ value (space for classes, semicolon for JS and CSS).