TYPO3.Fluid - Feature # 36410

Status:	New	**Priority:**	Should have
Author:	Florian Carstens	**Category:**	ViewHelpers
Created:	2012-04-21	**Assigned To:**	
Updated:	2012-08-13	**Due date:**	
Has patch:	No		
Subject:	Allow templates to send arguments back to layout		

Description

Layouts can differ a little on a website from page to page without affecting the whole layout. For example: Some pages contain a sidebar, others not. Or: Each page has it's own background which is set by a body or div container css class. I would like to set these kind of settings by sending arguments to the layout. That could e.g. be realised by extending the layout viewHelper:

```
<f:layout name="Master" arguments="{cssClass: blueBackground}" />
```

Of course, I can do this in the controller. But I think this wouldn't be the right place according to the MVC design pattern.

History

#1 - 2012-08-13 11:30 - Adrian Föder

I voted down for a simple reason: Sending back something removes the "side-effect-freeness" of Fluid Views (take it literally: they are **Views**). A View should be "read only", or "non-desctructive", or whatever you like.
For the same reason, by the way, I also agree to not having writable variables inside a view (like eg Smarty has).

#2 - 2012-08-13 11:37 - Alexander Berl

Good point. Do you have any ideas for alternative solutions to this problem?
I have similar issues, which are currently solved by including different stylesheets depending on the current view (can be done via ViewHelper or creating a <section> inside the layout <head>), but it's sometimes a hassle to have these sections in every template even if you don't need it.

#3 - 2012-08-13 11:46 - Adrian Föder

well, what solves most is having an own, "project specific" abstract class lying between the framework's own ActionController and your concrete action controllers:

```
abstract class AbstractBaseController extends \TYPO3\FLOW3\Mvc\Controller\ActionController
```

Then I have such one:

```
1  /**
2   * Initializes the view before invoking an action method.
3   *
4   * @param \TYPO3\FLOW3\Mvc\View\ViewInterface $view The view to be initialized
5   * @return void
6   */
7  protected function initializeView(\TYPO3\FLOW3\Mvc\View\ViewInterface $view) {
```

```
8       $localeCode = (string)$this->localizationService->getConfiguration()->getCurrentLocale();
9       $localeCode = str_replace('_', '-', $localeCode);
10      $this->response->setHeader('Content-Language', $localeCode);
11      $view->assign('localeCode', $localeCode);
12
13      $view->assign('account', $this->account);
14      $view->assign('bodyClass', $this->generateBodyClassName());
15  }
```

The generateBodyClassName() creates a smart representation of the current controller, representing entity, logged-in/logged-out status etc; and from here on most can be done via CSS.
take, for example,

```
<body class="ShopEntity ShopEntityOrderAction authenticated">
```

or similar (just an example).

So again, that abstract controller is pretty helpful so that I even encourage to do this as best-practice.