TYPO3.Fluid - Feature # 33117

Status:	Rejected	**Priority:**	Could have
Author:	Adrian Föder	**Category:**	ViewHelpers
Created:	2012-01-11	**Assigned To:**	Adrian Föder
Updated:	2012-05-08	**Due date:**	
Has patch:	Yes		
Subject:	returning plain \DateTime object in Format\DateViewHelper		

Description

The DateTime ViewHelper allows passing by a string that is converted to a DateTime object, which again is formatted against the requested pattern.

The current signature is

```
/**
     * Render the supplied DateTime object as a formatted date.
     *
     * @param mixed $date either a \DateTime object or a string that is accepted by \DateTime constructor
     * @param string $format Format String which is taken to format the Date/Time
     * @return string Formatted date
     * @api
     */
    public function render($date = NULL, $format = 'Y-m-d')
```

Providing an additional option "asObject" or such, that returns the \DateTime object without formatting, would allow the user to compare Dates with the IfViewHelper, because Date object could be compared with < or > each other (http://de2.php.net/manual/en/datetime.diff.php, Example #2).

So this should allow something like

```
<f:if condition="{model.date} < {f:format.date(date: 'now', asObject: 'true')}">
The date is due.
</f:if>
```

I know that this might not fit 100% into "formatting" and it additionally changes the return type via parameter (kind of "FlagArgument" http://martinfowler.com/bliki/FlagArgument.html), but maybe you're fine with it, because it's useful and the functionality is only almost one LOC away.

History

#1 - 2012-01-11 14:42 - Gerrit Code Review

- Status changed from New to Under Review

Patch set 1 for branch **master** has been pushed to the review server.
It is available at http://review.typo3.org/7737

#2 - 2012-05-07 16:16 - Robert Lemke

- Category set to ViewHelpers

I like the idea to be able to compare dates more easily, but I agree that by switching the returnValue by a flag and returning an object at all is a bit odd, because it servers – as far as I can see – only the purpose of making comparisons possible.

Would it be possible to enhance the comparison logic (ifViewHelper) instead so that it can recognize an RFC 2822 formatted date (http://www.faqs.org/rfcs/rfc2822.html)?

#3 - 2012-05-07 17:02 - Bastian Waidelich

- Status changed from Under Review to Needs Feedback

Robert Lemke wrote:

> *[...] I agree that by switching the returnValue by a flag and returning an object at all is a bit odd [...]*

I agree, this seems to be a pretty crude solution and I would be interested in a concrete use case

@Adrian so you have two dates in your view, one a \DateTime object, one a string or could you elaborate?

#4 - 2012-05-07 17:33 - Adrian Föder

Hi,
glad both of you found the time to have a look; Bastian, the concrete use case is, hopefully, described in the Fluid snippet above; I'll try to break it down. The sample reads

```
1<f:if condition="{model.date} < {f:format.date(date: 'now', asObject: 'true')}">
2The date is due.
3</f:if>
```

model.date is itself of type \DateTime, as usual. f:format.date could take 'now' as date, resulting to the current timestamp. With my "patch idea"

applied, the ViewHelper will return \DateTime representing *now*. With the above example than, you're able to in-line check whether a date is in the past, for example. Because PHP is capable of comparing two \DateTime objects just with the < or > operator (also see link above).

So, that's the use case. Those months ago, Sebastian generally liked the idea (see http://www.rvantwisk.nl/flow3-irc-log/2012/january/11.html 12:03 ff, the Internet doesn't forget ;-) but I also are not absolutely comfortable because of the Flag argument style...

#5 - 2012-05-07 19:07 - Bastian Waidelich

Adrian Föder wrote:

> *Bastian, the concrete use case is, hopefully, described in the Fluid snippet above; [...]*

Sorry, missed that one.
IMO the nicest way to get around this in your very case would be an additional getter in the model like

```
1public function isDue() {
2    return $this->date < new \DateTime();
3}
```

And then you could write

```
1<f:if condition="{model.due}">
2The date is due.
3</f:if>
```

This has two major advantages:

- less domain logic in the view
- more domain logic in the model

That way, if you need to change the way a due date is calculated (for example if [model] is due as long as it's on the same day) you only need to adjust it at one place.

If you can't change the model for some reason, I would still prefer to set the due date in the controller:

```
1$this->view->assign('dueDate', new \DateTime());
```

> *So, that's the use case. Those months ago, Sebastian generally liked the idea [...]*

Sometimes even Sebastian is wrong. Just kidding ;) Let me know if you are not satisfied by the above solution or come accross another use case!

#6 - 2012-05-08 08:57 - Adrian Föder

Hi Bastian,

I just checked my actual use case, and it's actually just one occurrence where extending the model would be sufficient and fine.

The question is: in my opinion, PHP's possibility to compare dates against each other just with the comparison operator AND the fact that this is directly usable with the IfViewHelper, really begs to be used.
My use case isn't the question here, the question is if it's imaginable to be used in many other cases.

That's just meant as an additional trigger to gamble around with the idea a bit; I'm absolutely fine and won't be latched ;-) if we decide to literally abandon it.

#7 - 2012-05-08 15:34 - Bastian Waidelich

Adrian Föder wrote:

Hi Adrian,

> *I just checked my actual use case, and it's actually just one occurrence where extending the model would be sufficient and fine.*

Thats great!

> *The question is: in my opinion, PHP's possibility to compare dates against each other just with the comparison operator AND the fact that this is directly usable with the IfViewHelper, really begs to be used.*

Right, but that's the case isn't it?

```
1 <f:if condition="{model.date} < {dueDate}">
```

works, if {dueDate} is a \DateTime, too.

> *My use case isn't the question here*

I don't think so. In order to avoid a "YAGNI mess" we should back up every new feature with a realistic use case.
I'm not saying that there is none for this feature - but as long as we don't know it we shouldn't introduce the feature in my opinion.

> *I'm absolutely fine and won't be latched ;-)*

Ok, great. Thanks for this ;)
I would now go ahead and abandon the change. We'll be able to restore it anytime if we come accross a real need!

#8 - 2012-05-08 16:13 - Adrian Föder

- Status changed from Needs Feedback to Rejected