TYPO3.Fluid - Feature # 33394

Status:	Needs Feedback	**Priority:**	Should have
Author:	Tobias Liebig	**Category:**	Core
Created:	2012-01-23	**Assigned To:**	Tobias Liebig
Updated:	2014-08-14	**Due date:**	
Has patch:	No		
Subject:	Logical expression parser for BooleanNode		

Description

In boolean ViewHelper attributes, like the f:if condition, one might want to use boolean expressions and boolean operators like && (and), || (or) or ! (not).

The BooleanNode needs to parse this logical expressions.

Related issues:

related to TYPO3.Fluid - Feature # 26665: Fluid: Implement String comparison	**Resolved**	**2010-03-10**
related to TYPO3.Fluid - Feature # 33215: RFC: Dynamic values in ObjectAccess...	**New**	**2012-01-16**
related to TYPO3.Eel - Feature # 39564: Eel Parser RegEx should support quote...	**Resolved**	**2012-08-07**
related to TYPO3.Eel - Feature # 38379: Implement a Eel-ViewHelper	**New**	**2012-06-25**
duplicated by TYPO3.Fluid - Feature # 40338: Make possible to combine conditi...	**Closed**	**2012-08-28**
blocks Core - Feature # 48221: BooleanNode must support logical OR/AND	**New**	**2013-05-15**

History

#1 - 2012-01-23 19:01 - Gerrit Code Review

- Status changed from New to Under Review

Patch set 2 for branch **master** has been pushed to the review server.
It is available at http://review.typo3.org/8614

#2 - 2012-05-07 17:06 - Robert Lemke

- Target version deleted (1.1)

#3 - 2012-05-07 17:14 - Robert Lemke

- Status changed from Under Review to Needs Feedback

#4 - 2012-05-07 17:18 - Tobias Liebig

Thanks for the reminder.

This change will be obsolete, if Eel is available in Fluid for FLOW3 and also in Fluid for extbase

#5 - 2012-06-18 20:21 - Jacob Floyd

How will Eel be integrated in Fluid? A view helper? I couldn't find a ticket about that - is there one? Do we need one?

If there is a ticket, and we are going to use Eel instead of implementing the logical operators within the boolean ViewHelper attributes, then this issue should depend on the other issue.

#6 - 2012-06-25 19:12 - Tobias Liebig

~~There is no other Issue/Ticket yet.~~ Just created two tickets.

IMHO we need to do two separate steps

- First of all Eel should be available within Fluid (the FLOW3-Package). I think the easiest way would be to implement a Eel-ViewHelper. (#38379)
- Then the Eel-Package (and this new VH) should be backported to extbase (and become a part of extbase) (#38380)

#7 - 2012-06-26 05:57 - Sebastian Kurfuerst

Actually Eel should be integrated into Fluid using the syntax of "${eel-expression}".

Greets, sebastian

#8 - 2012-06-26 08:30 - Tobias Liebig

Hej Sebastian,

what's the advantage of having a new build-into-fluid syntax (${..}) instead an eel viewhelper, except of being a little shorter? I think a viewhelper will be much easier to implement instead of extending the fluid template parser.

#9 - 2012-06-26 10:23 - Bastian Waidelich

Tobias Liebig wrote:

> *what's the advantage of having a new build-into-fluid syntax (${..})*

Can you think of an example and compare the syntax?
I'm not sure how that would look like in eel, but with a new ViewHelper it is probably rather technical:

```
1<f:if condition="{f:eel(query: 'foo == \'bar\'')}">
```

#10 - 2012-06-26 10:30 - Tobias Liebig

Bastian Waidelich wrote:

> *Tobias Liebig wrote:*
>
> > *what's the advantage of having a new build-into-fluid syntax (${..})*
>
> *Can you think of an example and compare the syntax?*
> *I'm not sure how that would look like in eel, but with a new ViewHelper it is probably rather technical:*
> *[...]*

i'd be totally fine with that, if it means its easier to implement, which actually means we could have it earlier in Fluid for extbase :-)

i guess in Sebastians suggestion it would be like:

```
1<f:if condition="${'foo == \'bar\'}">
```

#11 - 2012-06-26 15:26 - Sebastian Kurfuerst

Hey,

@Bastian: even easier:

```
<f:if condition="${foo == 'bar'}">
```

@Tobias: I'd rather not implement a half-baked solution which has poor usability -- rather change the Fluid parser for that.

Actually the eel parser contains already a RegEx for detecting eel expressions:
http://git.typo3.org/FLOW3/Packages/TYPO3.Eel.git?a=blob;f=Classes/Package.php

It's still very poor and needs to be dramatically improved, but once that is done Fluid can be extended quite easily to support that. However the improvement can be done test-driven.

Greets, Sebastian

#12 - 2012-06-26 15:46 - Bastian Waidelich

Sebastian Kurfuerst wrote:

> *@Tobias: I'd rather not implement a half-baked solution which has poor usability*

I agree. No problem in having an eel ViewHelper in some other extension but I'd not "ship" that with Fluid.
If implementing eel to the parser turns out to be too complex for now, we should at least support literal comparison (#6757)

#13 - 2012-08-06 12:14 - Alexander Berl

I'm not sure if it's a good idea to add a new syntax construct for expressions in fluid. It could confuse people on when to choose eel syntax "${ ... }" and when to choose standard fluid syntax "{ ... }".
If I'm not mistaken, eel could theoretically completely replace standard fluid syntax in functionality, right? So why not make it really replace the fluid internal expression parsing. Maybe the current parser could still be a fallback solution for when the eel package isn't available or not installed properly, but they should share the same fluid syntax IMO.

#14 - 2012-08-07 13:44 - Christopher Hlubek

I think to re-use the same expression parser is generally a great idea and was also a motivation when writing Eel. But there are some (minor?) problems we need to solve to make it a smooth replacement:

Object / Array access looks the same in Fluid and Eel, but there a some differences:
- Fluids ObjectAccessorNode does support access to public getters (getFoo() and isFoo()) and public properties (for objects, array keys (for arrays and ArrayAccess).
- Eel allows to call just about any public method of an object, which could be unwanted in Fluid (imagine a destructive method being called in the view). So we need some method whitelisting or language feature support (e.g. disable method calls in Fluid Eel) here to make it safe.
- Additionally we need to replicate the ObjectAccessorNodes behaviour when accessing properties to make Eel a replacement in Fluid. But that should be rather trivial.

I think using the same syntax could cause some harder edge cases, imagine building arrays for example:

```
<f:if condition="{blog.name == 'My blog'}">
<f:link.action ... arguments="{blog: post.blog}">My blog</f:link.action>
</f:if>
```

Syntax-wise the arguments array looks like an Eel expression, but it is handled by an ArrayNode in Fluid. For an Eel Expression, it has to be written as arguments="{{blog: post.blog}}" or we introduce a Ruby 1.9 like shorthand "object" / array notation: arguments="{blog: post.blog}". But there are a lot of templates in the wild, where something like arguments="{name: '{f:some.viewhelper(arg: \"Foo\"})'" is used. Right now I think this works in Fluid because it just uses Regular expressions and recursively parses the template (Sebastian?).

To make it consistent we would need to replace everything but the shorthand syntax in Fluid with Eel expressions.

Alexander Berl wrote:

> *If I'm not mistaken, eel could theoretically completely replace standard fluid syntax in functionality, right? So why not make it really replace the fluid internal expression parsing. Maybe the current parser could still be a fallback solution for when the eel package isn't available or not installed properly, but they should share the same fluid syntax IMO.*

#15 - 2013-02-21 00:24 - Stefan Neufeind

Any chance to get this going again maybe? I think it would be a huge step forward and would appreciate it.

#16 - 2013-04-16 16:08 - Alexander Berl

Christopher Hlubek wrote:

> *Object / Array access looks the same in Fluid and Eel, but there a some differences: - Fluids ObjectAccessorNode does support access to public getters (getFoo() and isFoo()) and public properties (for objects, array keys (for arrays and ArrayAccess).*
> *- Eel allows to call just about any public method of an object, which could be unwanted in Fluid (imagine a destructive method being called in the view). So we need some method whitelisting or language feature support (e.g. disable method calls in Fluid Eel) here to make it safe.*

As far as I see, the method calling is controlled by the Context. So would it be a possible solution to create a "FluidContext" which throws an exception in its call method?
Then within Fluid if Eel is available, a new EelEvaluator is created as well as a FluidContext and all expressions evaluated that way.

> *- Additionally we need to replicate the ObjectAccessorNodes behaviour when accessing properties to make Eel a replacement in Fluid. But that should be rather trivial.*

Isn't the ObjectAccessor already used in the Eel Context::get() method, which afaik is responsible for all non-method call access (i.e. public properties

and everything accessible via getters).

I'm not sure when/where the ${} syntax is needed though... also the array notation has to be checked, but I can't imagine that being a big problem.

#17 - 2013-05-15 16:29 - Stefan Neufeind

Just a small ping ... Anything we can help with maybe?

#18 - 2013-05-16 08:46 - Tobias Liebig

I didn't put any effort into this patch as it seems like eel should solve the issue.
No idea about if and when eel is available in Fluid and backported to extbase too.

#19 - 2013-05-16 14:39 - Alexander Berl

Stefan Neufeind wrote:

> *Just a small ping ... Anything we can help with maybe?*

You're free to try working out a solution to the problems mentioned in using Eel inside Fluid, maybe with the hints I gave it will be a good starting point. Anything you find will be valuable information and maybe even a good solution turns out that can be used directly

#20 - 2013-05-19 14:55 - Alexander Berl

Just some update, as I did some more digging into the problem:

My basic idea would be to create an EelExpressionNode in Fluid, which evaluates by creating a Fluid Eel Context and an Eel Evaluator, then sending the expression through that. The CompilingEvaluator could be nicely used with the Template Compiling.

However, the main problem is one that Christopher already brought up:

```
arguments="{name: '{f:some.viewhelper(arg: \"Foo\"})'"
```

The problem here is, that we cannot simply put the whole arguments string as Eel expression, as Eel doesn't understand viewhelpers.
A solution could be, to parse out all shorthand viewhelpers inside attributes, create their own ViewHelperNode that assigns the result to a new variable, then replace the shorthand syntax inside the attribute string with the variable and put it all into the Eel expression.

Also, with that approach, all the need for ${} or {{}} array syntax is leviated, as the evaluator only works with the inner part of this Eel syntax anyway. As a benefit, the dynamic array index access syntax somearray[some.variable] would work out of the box. I therefore put the issue for that as related.

#21 - 2013-05-24 11:18 - Tymoteusz Motylewski

The lack of condition combinations in Fluid is one of the biggest pains in the daily development.
From Extbase/TYPO3 CMS perspective it would make more sense to first implement non-eel solution, so we don't have to wait until eel stabilizes and then backport it. This way we could include it in the next 6.2LTS release. See sample implementation of combining conditions from Claus Due here

https://github.com/NamelessCoder/vhs/blob/master/Classes/ViewHelpers/ConditionViewHelper.php
What do you think? Are you willing to merge such a change if I propose it? Or is it not an option?

#22 - 2013-09-06 16:12 - Dave no-lastname-given

It seems to me that waiting for the Eel dependency is dragging this issue into the nether-world. Yes Eel may support reg-expr, but in my mind the need for basic comparators is more critical than support for regular expressions.

Maybe the solution should be to support the basic comparators first, add Eel support later.

A conditional is a core operator, even in PHP itself an "if" statement does not support regular expressions, only comparator operators. I think eel should be it's own Viewhelper and it should NOT be a dependency for the conditional Viewhelper. It is creating an unnecessary layer of bloat if you dont need to use reg-expr at all.

In my mind, all these issues can be easily solved if all ViewHelper output can be assigned to a template variable, as I suggested in [[http://forge.typo3.org/issues/48355]] and as I further describe in the same issue; if Fluid is opened up to the whole PHP API you don't even need Eel as <f:preg_match /> and any other PHP function would be available to Fluid.

By doing this it would eliminate the number of Core Viewhelpers that need to be written, and it would speed up template parsing by using PHP directly and not loading zillions of viewhelpers that replicate PHP functionality.

In the end I feel these discussions of "missing features" stifle Fluid development and create unnecessary constraints and convulations, when you already have the perfect templating API in PHP. It is there, underneath everything, and IMHO it is shamefully not accessible from a template.

It seems like Fluid is doomed to reinvent the wheel.

#23 - 2014-07-04 10:48 - Mathias Brodala

What's the status on this one? Either this patch should be integrated or Eel expressions made available in Fluid, also thinking about how to implement this in TYPO3 CMS Fluid.

#24 - 2014-08-14 12:56 - Dave no-lastname-given

The nether-world is a comfortable place. ;-)

I still really think the whole PHP API should be opened up to Fluid. It would solve all these unnecessary problems. I have not found or heard if there is a discussion anywhere about this topic. I was also told that there is a design "philosophy" for Fluid but I have not seem to have found the corresponding document or discussion. I do not think either exists.

I suspect so much effort has been put into Eel, that some or someone does not want to have it made obsolete. In my personal opinion, Eel integration would just create another unnecessary abstraction on top of another abstraction (Fluid) which is built upon another abstraction (PHP). I also suspect for some it is hard to accept that Eel is code bloat. It just adds another level of unnecessary parsing.

In my mind Eel was created as a "jQuery" like solution to create some elegancy in TypoScript to problems that were already being solved by custom user functions. In Fluid, Eel is a solution looking for a problem. IMHO the extra processing overhead it generates in templating is illogical.

jQuery was invented because it solved a real problem of elegantly manipulating HTML DOM where there was so many different and ugly ways to do it across the various browser implementations. In Typo3 there is no such variance, the underlying implementation (PHP) is a constant.

I would understand the need for Eel if Typo3 was built using a mix of PHP, Ruby, Java, Haskell and Python and you needed a common interface

between them all. Is there a Haskell port of Typo3? I didn't think so...

#25 - 2014-08-14 13:21 - Marc Neuhaus

FYI, i created a changeSet for TYPO3.Fluid to integrate Eel into the parser 2 weeks ago: https://review.typo3.org/#/c/31707/

Regarding the added complexity: i agree that is sometimes a problem, yet in this case eel itself has been proven to be really sturdy with boolean expressions, etc.

Regarding the performance: The integration in that changeSet uses the CompilingEelParser, which produces real php code that is cached as real php code inside the fluid template cache.

Cheers
Marc

#26 - 2014-08-14 14:10 - Dave no-lastname-given

But Eel does not solve all the other problems, it just adds to the bloat. Where as by opening up the entire PHP API to Fluid it eliminates most of them. Not just boolean expressions but basically anything PHP can do. All it takes to get the **entire** PHP API is a little reflection, by asking if a native function or method exists and if it does call it as if it was a ViewHelper and return the output.

Surely reinventing the wheel is not part of the Fluid design philosophy?