TYPO3.Flow - Feature # 4668

Status:	Rejected	**Priority:**	Should have
Author:	Nikolas Hagelstein	**Category:**	Validation
Created:	2009-09-17	**Assigned To:**	Karsten Dambekalns
Updated:	2010-10-20	**Due date:**	
PHP Version:			
Has patch:			
Complexity:			
Subject:	Validation improvement		

Description

Sometimes it makes sense to stop validation after the first failure.
E.g.
@validation NotEmpty, EmailAddress

In this case it makes no sense to face the user with both errorMessages.

Though this can be resolved with a custom ViewHelper it would be nice if there would be a suitable annotation.

History

#1 - 2009-09-28 15:17 - Karsten Dambekalns

- Status changed from New to Needs Feedback

- Assigned To set to Karsten Dambekalns

You write "sometimes it makes sense" - how would you imagine defining **if** it makes sense or not?

#2 - 2009-09-28 15:55 - Nikolas Hagelstein

Karsten Dambekalns wrote:

> *You write "sometimes it makes sense" - how would you imagine defining **if** it makes sense or not?*

This decision needs to be taken by the developer by e.g. introducing a new notation something like:
@validation NotEmtpy:StopOnFail, EmailAddress ... or similar.

#3 - 2009-09-28 16:22 - Falk Kühnel

Should the validation not interrupt after the first invalid test anyway? Why would you test the others aswell?

#4 - 2009-09-28 16:50 - Karsten Dambekalns

Falk Kühnel wrote:

> *Should the validation not interrupt after the first invalid test anyway? Why would you test the others aswell?*

Well, I have been upset on quite some websites throughout my life that tell me to enter a password with only 8 chars, and only upon trying again inform me that I should only use ASCII chars as well. You get the point!? Of course that **should** be solved by a good UI in the first place, but it might make sense to display all errors. Or not?

#5 - 2009-09-28 16:54 - Karsten Dambekalns

Nikolas Hagelstein (pulponair) wrote:

> *This decision needs to be taken by the developer by e.g. introducing a new notation something like:*
> *@validation NotEmtpy:StopOnFail, EmailAddress ... or similar.*

Hm, that seems rather longish. Is the following too compact?

Perform all evaluations:
@validate NotEmpty, EmailAddress, FooBar

Stop after any error:
@validate NotEmpty; EmailAddress; FooBar

Perform the first two in any case, FooBar only if the forst two pass:
@validate NotEmpty, EmailAddress; FooBar

#6 - 2009-09-28 17:09 - Falk Kühnel

Karsten Dambekalns wrote:

> *Well, I have been upset on quite some websites throughout my life that tell me to enter a password with only 8 chars, and only upon trying again inform me that I should only use ASCII chars as well. You get the point!? Of course that **should** be solved by a good UI in the first place, but it might make sense to display all errors. Or not?*

True in these cases, but my personal default would be to stop validation if the first one fails. I personally don't care for all errors, but rather speed up the validation process.
The case you are referring to is annoying (and should be explained on the form page, before you enter your data), but i would not like to see an error message, that consist of 8 lines of errors, since you used 8 validators. In those cases, a new Validator should be written, that gives a proper error message and makes use of the previously mentioned validators.

Nikolas Hagelstein (pulponair) wrote:

> *Hm, that seems rather longish. Is the following too compact?*

Thats too compact. In favor of readability i would restrain from such a syntax. I cant even count the times i haven't found an error, just because i put a semicolon instead of a comma :)

#7 - 2009-09-28 17:21 - Bastian Waidelich

I agree to Karsten, we should keep the option to collect all validation errors at once.
What do you think of following syntax:

@validate NotEmpty, EmailAdress, Unique

vs

@validate NotEmpty, EmailAdress
@validate Unique

..Though - the current implementation seems to do the job - If you only want to show the first error, that's up to you (currently by writing your own error view helper e.g.)

#8 - 2009-09-29 09:17 - Falk Kühnel

Bastian Waidelich wrote:

> *I agree to Karsten, we should keep the option to collect all validation errors at once.*

Having the option is always good :)

> *What do you think of following syntax:*

I like it. Seems clean and easy to understand.

#9 - 2009-09-29 16:15 - Karsten Dambekalns

- Status changed from Needs Feedback to Rejected
- Target version set to 1.0 alpha 5

Bastian Waidelich wrote:

> *What do you think of following syntax:*
> *[...]*
> *vs*
> *[...]*

We cannot rely on the order of annotations. Although they seem to end up in reflection the same way as in the source code, there is no guarantee for that. Thus this is not an option.

In the example above one would simply leave out NotEmpty, as EmailAddress implies NotEmpty and for anything more complex a custom validator is better anyway. And the time spent with validation is not an issue, so this becomes a presentation problem, which should be solved in the responsible view helper(s) (they should be smart enough to reformat the error messages of the involved validators into something like "hey, you moron, an empty string really is no valid email address").

> *..Though - the current implementation seems to do the job - If you only want to show the first error, that's up to you (currently by writing your own error view helper e.g.)*

To keep the project on track we need to avoid catering for special needs at the moment. For now this should be solved by doing the right thing in your view or with a custom validator.

#10 - 2009-10-19 16:27 - Nikolas Hagelstein

Karsten Dambekalns wrote:

> *In the example above one would simply leave out NotEmpty, as EmailAddress implies NotEmpty and for anything more complex a custom validator is better anyway. And the time spent with validation is not an issue, so this becomes a presentation problem,*

You are right for this "emailAddress" sample but it will quickly become a matter performance if you e.g. got a custom validator checking a value against a webservice, complex db call or performing any other kind of expensive check.

ATM the only way to get arround this is to leave simple (i.e. syntactic ones) validations to the framework magic and perform complex validations within your target action, manually generate an error object, manipulate the request and forward it back to the origin action.

If anyone is aware of a more elegant way, please enlight me :).