TYPO3.Fluid - Feature # 43457

Status:	Closed	**Priority:**	Should have
Author:	Bastian Waidelich	**Category:**	View
Created:	2012-11-29	**Assigned To:**	Bastian Waidelich
Updated:	2013-05-07	**Due date:**	
Has patch:	No		
Subject:	Cacheable partials		

Description

Idea: add a new argument to the **render** ViewHelper that allows to cache a partial:

```
1  <f:render partial="SomePartial" cache="true" />
```

The cache identifier should be created based on a combination of the unique partial identifier & it's arguments.

Note: Sections could be cacheable, too

Related issues:

related to TYPO3.Fluid - Feature # 3291: Cacheable viewhelpers	**Needs Feedback**	**2009-05-14**

History

#1 - 2012-11-29 17:03 - Gerrit Code Review

- Status changed from Accepted to Under Review

Patch set 1 for branch **master** has been pushed to the review server.
It is available at http://review.typo3.org/16856

#2 - 2012-11-29 17:46 - Adrian Föder

since it's common practice, meanwhile, to assign variables per default to every view, it's likely to happen that arguments are not explicitly passed by via attributes and are just expected to be present in the "target" partial (I, for example, have the authenticated {party} present everywhere).

In this case, one would provide the "argument" attribute anyways in order to influence the caching, right?
For example,

```
1<f:render partial="LastLogin" cache="true" arguments="{party: party}" />
```

...even if the {party} would be present in the partial anyways.

Is that correct?

Additionally, what about a, for example, vary attribute (like the Vary HTTP header) that expects a set of implemented variabilities, for example the authenticated user or the current language? This could be, then

```
1<f:render partial="LastLogin" cache="true" vary="{0: 'Account', 1: 'CurrentLanguage'}" />
```

Note that Account and CurrentLanguage are keyword here which trigger a Handler maybe that provides the appropriate cache influencer (for Account that would maybe be the AccountIdentifier, and for CurrentLanguage that would be de-DE or the masterset (de) or so...

aaaaaaaand does an expiration attribute make sense here, would that be the right place? This could either be a timestamp or even the keyword "session" for example.

But I see, I'm adding functionality over functionality again here :)

#3 - 2012-11-29 17:53 - Bastian Waidelich

Adrian Föder wrote:

> *In this case, one would provide the "argument" attribute anyways in order to influence the caching, right?*
> *For example,*
>
> *...even if the {party} would be present in the partial anyways.*

How would {party} be present in the partial anyways?
Only settings and passed arguments are available.

> *Additionally, what about a, for example, vary attribute (like the Vary HTTP header) that expects a set of implemented variabilities,*
> *for example the authenticated user or the current language?*

At first sight I think that would make it too complex. Instead if you want to make the partial depending on the last login time, you could do

```
1<f:render partial="LastLogin" arguments="{lastLogin: user.lastLogin}" cache="true" />
```

or

```
1<f:render partial="LastLogin" arguments="{user: user, lastLogin: user.lastLogin}" cache="true" />
```

to make it depending on user **and** last login time.

> *aaaaaaaand does an expiration attribute make sense here, would that be the right place?*
> *This could either be a timestamp or even the keyword "session" for example.*

Mh.. Same argument, though it would make sense of course..

#4 - 2012-11-30 09:54 - Benno Weinzierl

I like this solution very much!

Would it make sense to have a config where you can bypass the arguments-check?
I dont know how much performance it will cost if there are large arrays/collections inside?

My usecase is this:
I generate a tree of categories by using getChildren() on the parent category. This generates several sql-queries and therefore i want to cache it.
In this case i know exactly then a category is changed ant would prefere to clear the cache automatically when a category is changed/deleted/created.

I will try it out at hte weekend and tell you my experience with it.

#5 - 2012-11-30 11:19 - Bastian Waidelich

Benno Weinzierl wrote:

> *I like this solution very much!*

Thank you ;)

> *Would it make sense to have a config where you can bypass the arguments-check?*
> *I dont know how much performance it will cost if there are large arrays/collections inside?*

Yes, that would make sense.
I don't think that performance is a big problem (performance tests to come) but I had issues with this: e.g. I use some "data transfer object" (w/o identifier) to render a menu partial.
This will throw an exception because no cache identifier could be created from the object..

I'm thinking about omitting the arguments for the cache identifier and instead add another argument "cacheVariables" or similar:

```
1<f:render partial="Foo" arguments="{has: 'noeffect'}" cache="true" cacheVariables="{has: 'aneffect'}" />
```

What do you think?

#6 - 2012-12-02 13:41 - Benno Weinzierl

Bastian Waidelich wrote:

> *I don't think that performance is a big problem (performance tests to come) but I had issues with this: e.g. I use some "data transfer object" (w/o identifier) to render a menu partial.*
> *This will throw an exception because no cache identifier could be created from the object..*

I did a little testing of your solution yesterday and there is another problem with objects which implement IteratorAggregate like i my case Doctrine\ORM\PersistentCollection. They are not handled right by AbstractTemplateView::convertObjectsToHashes(). They are handeld as objects (width no identifier) and not as arrays.

I did change it in my local working copy but my solution is not realy usable in real live it only shows the general problem.

> *I'm thinking about omitting the arguments for the cache identifier and instead add another argument "cacheVariables" or similar:*

Because i think it is not possible to handle every possible argument (large collections, objects with no identifier and so on) the cacheVariables

approach may be better.
Maybe it is better to call it cacheIdentityVariables as they just define the cache identity?

Another question:
If a persisted entity is changed the cache sould be cleared. How would that be handled? In my case the only solution would be a manual cache clean implemented by the developer. He would need to call a clearcache-method with the same cacheIdentityVariables?
In my case there is no way to determine the need to change the cache automatically because i just provide a collaction of categories and the active category to the partial (to generate a menu). In the partial i call getChildren which fetches completely new objects not known at the moment the partial is called.

#7 - 2012-12-12 16:12 - Bastian Waidelich

- Target version set to 2.1

Benno Weinzierl wrote:

Hi Benno, thanks for your feedback!

> *Bastian Waidelich wrote:*
>
> > *I did a little testing of your solution yesterday and there is another problem with objects which implement IteratorAggregate like i my case Doctrine\ORM\PersistentCollection.*
> > *They are not handled right by AbstractTemplateView::convertObjectsToHashes(). They are handeld as objects (width no identifier) and not as arrays.*

Yes, that's right.. With #42715 we can properly serialize Collections it seems. That might be a solution..

> *I'm thinking about omitting the arguments for the cache identifier and instead add another argument "cacheVariables" or similar:*
>
> *Because i think it is not possible to handle every possible argument (large collections, objects with no identifier and so on) the cacheVariables approach may be better.*

Yes, that's right. Besides, by default, all settings of the current package are added to the arguments by the render ViewHelper.

> *Maybe it is better to call it cacheIdentityVariables as they just define the cache identity?*

Yes, I like that!

> *If a persisted entity is changed the cache sould be cleared. How would that be handled?*

Absolutely. This is another issue we discussed recently. We'd also need that for #29972 and some related features..
One approach is to use a generic service to determine a cacheIdentifier for an object (see https://github.com/mneuhaus/Community.CacheExtensions/blob/master/Classes/Community/CacheExtensions/Services/CacheIdentityService.php for a draft from Marc Neuhaus) and then we probably need some generic callbacks for when an object is updated/removed in order to flush related caches and such..

This won't make it into 2.0 unfortunately, but it won't be long until 2.1 I hope ;)

#8 - 2013-01-08 18:11 - Alexander Berl

Actually, I prefer the CacheViewHelper solution, as it is much more flexible and allows for whole templates or just small snippets to be cached.
As for the identifier, would using a mix of the current template/partial name + count be a solution? Not sure if that's possible at all, haven't been into FLOW code recently, but that idea just came up.
Alternatively, something similar to ruby could work, where you provide the cache identifier suffix:
http://guides.rubyonrails.org/caching_with_rails.html#fragment-caching

#9 - 2013-05-07 18:44 - Bastian Waidelich

- Status changed from Under Review to Closed

- Target version deleted (2.1)

Change has been abandoned (see https://review.typo3.org/#/c/16856/ for details).