TYPO3.Flow - Feature # 27329

Status:	Rejected	**Priority:**	Should have
Author:	Bastian Waidelich	**Category:**	Persistence
Created:	2011-06-09	**Assigned To:**	Karsten Dambekalns
Updated:	2011-09-30	**Due date:**	
PHP Version:			
Has patch:			
Complexity:			
Subject:	Wrap Doctrine ArrayCollection in some FLOW3 collection		

Description

FLOW3 currently uses the ArrayCollection of Doctrine by default to store multivalue properties in models. It comes with nice features such as **map()** and **filter()**.
I suggest to wrap this in a custom FLOW3 collection in order to be able to extend its functionality and to gain more abstraction from Doctrine.
Besides this would make it easier to backport this feature to Extbase so that people already use the "correct" type in their models.
e.g.

```
1class \F3\FLOW3\Utility\Collections\ArrayCollection extends \Doctrine\Common\Collections\ArrayCollection {
2}
```

@Karsten could you share your opinion on this?

History

#1 - 2011-09-05 16:37 - Robert Lemke

- Target version set to 1.0 beta 2

+1 for the idea.

How about naming it \TYPO3\FLOW3\Property\DataType\ArrayCollection ?

#2 - 2011-09-05 16:44 - Karsten Dambekalns

I don't see a benefit in wrapping it. The Collection interface is self-contained and wrapping it would only serve to the "not-invented-here" syndrome.

If you all think we should do it, though, fine with me.

#3 - 2011-09-05 17:50 - Bastian Waidelich

Karsten Dambekalns wrote:

> *I don't see a benefit in wrapping it. [...]*

We had this discussion recently. But I still think, that it would make sense to - at least - use a FLOW3 interface to code against.

I can already think of a concrete reason: If there was a public method that marks the collection "dirty", we could get around the (somewhat hacky)

```
$sessionTypeIndex = $this->sessionTypes->indexOf($sessionType);
$this->sessionTypes->set($sessionTypeIndex, $sessionType);
```

for entities that are not bound to a repository (#13324) and instead do something like:

```
$this->sessionTypes->update($sessionType);
```

(I know, ArrayCollection::initialize() would probably do the same, but...)

#4 - 2011-09-09 10:21 - Karsten Dambekalns

- Target version changed from 1.0 beta 2 to 1.0.0

One issue would be that we'd deviate from the "your Doctrine knowledge can be applied as is". And to me it's still only one more wrapper that wouldn't really add functionality, if we put something like updateElement() in Doctrine itself.

But even that is only a "workaround" for our failure to adhere to our own principles (only access by traversal in such cases).

#5 - 2011-09-09 11:15 - Bastian Waidelich

Karsten Dambekalns wrote:

> *One issue would be that we'd deviate from the "your Doctrine knowledge can be applied as is".*

On the other hand we would be in control of the API again - so we could make sure that it stays in sync if we think it makes sense.

> *[...] if we put something like updateElement() in Doctrine itself.*
> *But even that is only a "workaround" for our failure to adhere to our own principles*
> *(only access by traversal in such cases).*

You might be right, and I'm not an DDD expert. But in the concrete example - what could we do better in order to avoid the (not very DDD like)

```
$sessionTypeIndex = $this->sessionTypes->indexOf($sessionType);
$this->sessionTypes->set($sessionTypeIndex, $sessionType);
```

Or do you reccon, that the domain model of the conference is not DDD style in the first place? (I'm not beeing sarcastic)

#6 - 2011-09-30 12:16 - Robert Lemke

+1 for introducing a FLOW3 interface / wrapper for collections

#7 - 2011-09-30 13:44 - Karsten Dambekalns

- Status changed from Needs Feedback to Rejected

I gave this more thought yesterday and today. My conclusion: if we do this, we'll break things or gain nothing.

our own interface

We can add our own interface extending the original one. Gives a nice feeling but breaks as soon as something is involved that is coming from Doctrine and "only" implementing *their*Collection interface.

our own ArrayCollection

We can add our own ArrayCollection extending theirs. This does not help with the "dirty-marking", as that in involves instances of PersistentCollection - not ArrayCollection anymore, and not at all our own ArrayCollection. Adding functionality would be possible, but anything reconstituted would again only be PersistentCollection. Thus the API of Doctrine's Collection interface is the limit.

So, this doesn't work. Sorry.