TYPO3.Flow - Feature #6609

Implement joins for queries

2010-02-25 19:09 - Andreas Förthner

Status:	Closed	**Start date:**	
Priority:	Should have	**Due date:**	
Assignee:	Karsten Dambekalns	**% Done:**	0%
Category:	Persistence	**Estimated time:**	0.00 hour
Target version:			
PHP Version:		**Complexity:**	
Has patch:	No		

Description

One possibility for a join implementation could work as follows:

First a precondition has to be considered: You are only allowed to join objects that have a relation to each other.

I.e. a person may have several e-mail-adresses. A possible join could now be, to load a person from the repository and directly load (join) all e-mail-adresses with the tld ".de" this person has onto the person object. The technical effect would be that the person is not loaded with all her e-mail-adresses but only with the joined ones.

Of course this is conceptually not correct, because the person in fact might have other addresses that would not be there by executing this join query. There are two solutions for this:

1. If you write a join query, you have to be aware that the result is only a part of the truth, meaning there could be more objects in the aggregate you didn't get by your query.
2. All objects you didn't get directly (didn't match the join condition) are lazy loaded if you want to access them at a later point. This would need a lazy loading mechanism that can lazy load collections (array, SplObjectStorage) on a per element basis.

To conclude the whole story. Joins in the object world can be used to reduce the objects that have to be loaded at a time. Meaning they are more like a optimization than a conceptual query construct.

Related issues:

Related to TYPO3.Flow - Feature #8774: Query::execute() should return a Proxy...	**Resolved**

History

#1 - 2010-03-05 16:19 - Andreas Förthner

- Assignee set to Karsten Dambekalns

We should go for solution 1.

#2 - 2010-03-08 17:45 - Jochen Rau

There is a great danger of data loss, if only a part of the related objects are loaded. Let's assume, that you only load the tree latest posts. Then we programmatically add a new post $blog->addPost($post). At persist time the $posts is detected being dirty and it gets persisted as it is with only 4 posts. This can be solved but makes it hard to determine the difference between the clean state and the dirty state.

In a first step, we should enable joins to restrict the result set of a query for aggregate roots. Let's assume we are in the OfferRepository. It would be very handy to be able to say

$query->greaterThanOrEqual('ageRange.minimumValue', $age)

where the property ageRange of the Offer holds an AgeRange object having a property minimumValue.

Jochen

#3 - 2010-03-10 09:13 - Jochen Rau

I will implement the object accessor syntax as an experimental feature in Extbase (#6755) to get feed-back on this feature.

#4 - 2010-07-09 14:20 - Robert Lemke

- Target version set to 1.0 alpha 11

Discuss, solve or close?

#5 - 2010-07-09 16:57 - Karsten Dambekalns

- Status changed from New to Accepted

- Start date deleted (2010-02-25)

Needs to be done anyway.

#6 - 2010-08-19 11:34 - Karsten Dambekalns

- Target version deleted (1.0 alpha 11)

#7 - 2012-03-12 17:48 - Christian Müller

- Status changed from Accepted to Closed

- Has patch set to No

With Doctrine we have JOINS, so this is closed.