TYPO3.Flow - Bug # 13324

Status:	Resolved	**Priority:**	Must have
Author:	Bastian Waidelich	**Category:**	Persistence
Created:	2011-02-24	**Assigned To:**	Karsten Dambekalns
Updated:	2011-09-07	**Due date:**	
PHP Version:			
Has patch:			
Complexity:			
Affected Flow version:			
Subject:	It's not possible to update entities without repository		

Description

Adding/removing entities to/from a collection of the parent object is possible by using the attach()/detach() methods of the SplObjectStorage.
But there is no equivalent for **updating** objects yet.

More specific: How could one update a comment of the BlogExample without creating a CommentRepository.

Related issues:

related to Conference Management - Task # 27629: Adjust Models to enable editing	**Resolved**	**2011-06-21**
related to Conference Management - Task # 27566: Streamline Conference editin...	**Resolved**	**2011-06-20**
related to TYPO3.Flow - Feature # 28486: Don't clone, use explicit change tra...	**Resolved**	**2011-08-17**

History

#1 - 2011-02-24 10:41 - Bastian Waidelich

in the repository updating is done like
1if ($this->persistenceManager->getIdentifierByObject($existingObject) !== NULL) {2$this->persistenceManager->replaceObject($existingObject, $newObject);3 [...]

in the replaceObject() method.
But the repository also keeps track of added and removed objects.

Maybe we need to work with a custom SplObjectStorage like in Extbase?

#2 - 2011-02-24 11:44 - Karsten Dambekalns

No update needed. If you change a persisted object (attached to anything persisted), changes are picked up automagically. No need for an update. If that does not work it's a bug, but we don't need anything new here!

#3 - 2011-02-24 12:04 - Bastian Waidelich

Karsten Dambekalns wrote:

> *No update needed.*

How would you solve this:

```
1public function updateAction(Comment $comment) {
2  ...
3}
```

usually you would do $this->commentRepository->update($comment);
So I'd expect something like $post->update($comment);

FYI i solved this for now by introducing following method to the abstract base controller:

```
 1/**
 2 * Replaces an existing object with the same identifier by the given object
 3 *
 4 * @param object $modifiedObject The modified object
 5 */
 6public function update($modifiedObject) {
 7    if ($modifiedObject->FLOW3_Persistence_isClone() !== TRUE) {
 8        throw new \F3\FLOW3\Persistence\Exception\IllegalObjectTypeException('The object given to update() was not a clone of a persistent object.', 1298540772);
 9    }
10    $uuid = $this->persistenceManager->getIdentifierByObject($modifiedObject);
11    if ($uuid === NULL) {
12        throw new \F3\FLOW3\Persistence\Exception\UnknownObjectException('The "modified object" does not have an existing counterpart in this repository.', 1298540861);
13    }
14    $existingObject = $this->persistenceManager->getObjectByIdentifier($uuid);
15
16    if ($this->persistenceManager->getIdentifierByObject($existingObject) === NULL) {
17        throw new \F3\FLOW3\Persistence\Exception\UnknownObjectException('The "existing object" is unknown to the persistence backend.', 1298540871);
18    }
19    $this->persistenceManager->replaceObject($existingObject, $modifiedObject);
20}
```

#4 - 2011-02-25 02:26 - Jochen Rau

I would agree with Karsten here: no change needed. Otherwise it would break the aggregate boundaries and thus the idea of atomic domain transactions. There are two options to handle an incoming Comment:

1. find the corresponding Post (through a method of the PostRepository) and update the Comment through a public function of this Post, or
2. treat the Comment as Aggregate Root and implement a CommentRepository (Comment Aggregates within a Post Aggregate doesn't break anything IMO).

#5 - 2011-02-25 09:09 - Bastian Waidelich

Jochen Rau wrote:

> *I would agree with Karsten here: [...]*

> *[...] update the Comment through a public function of this Post*

How? That was the point of this issue ;)
Of course my "solution" is just a hack to get it working for me. We needed s.th. like $this->comments->update($modifiedComment); but SplObjectStorage can't do that obviously..
We probably need a specialized "entity collection" that extends SplObjectStorage and knows how to find the original object by the modified clone..

#6 - 2011-02-25 10:23 - Karsten Dambekalns

In theory update() should take care of that. It recursively checks for entities that are clones and should update/replace them as well.

#7 - 2011-02-25 11:18 - Bastian Waidelich

Karsten Dambekalns wrote:

> *In theory update() should take care of that. [...]*

..but there is no update() in \SplObjectStorage ;)

#8 - 2011-03-21 12:08 - Karsten Dambekalns

The correct solution when taking the domain model into account would be to fetch the Post the comment is attached to and map the new values onto the Comment referenced by this Post. In other words: the problem is that the comment is completely detached from it's surroundings in this scenario...

#9 - 2011-06-10 13:01 - Bastian Waidelich

- Status changed from New to Needs Feedback

Karsten Dambekalns wrote:

> *The correct solution when taking the domain model into account would be to fetch the Post*
> *the comment is attached to and map the new values onto the Comment referenced by this Post.*
> *In other words: the problem is that the comment is completely detached from it's surroundings*
> *in this scenario...*

So, it's not solvable at all??

I need this to work for the conference site atm and with the FLOW3 refactoring my work around from above does not work anymore..
The case is even simpler, cause it's a 1:1 relation:
I just want to update the location of the current conference, so I tried

```
1public function updateAction(\F3\Conference\Domain\Model\Conference\Location $location) {
2  $this->conference->setLocation($location);
```

```
3}
```

But I end up with a new Location in the database and $this->conference->getLocation() returns NULL.

help

#10 - 2011-06-10 14:18 - Bastian Waidelich

After digging a bit deeper into this issue, here my findings:
The problem seems to be that \Doctrine\Common\Persistence\ObjectManager::contains() returns FALSE for a cloned object!?
\Doctrine\ORM\UnitOfWork::getEntityIdentifier() returns a UUID instead of the expected identitye (array('title' => 'Location Title')).

For now, i fixed it in the Conference package by writing:

```
1$this->persistenceManager->merge($modifiedObject);
```

But I think, that the framework should handle $this->conference->setLocation($modifiedLocation) automatically..

#11 - 2011-06-17 16:23 - Karsten Dambekalns

- Assigned To set to Karsten Dambekalns

Ok, for the conference when editing intermission types it's somewhat clear: editing the duration works, editing the title doesn't. That's because the title is part of the identity, and changing this is non-trivial at best. Because the system cannot know that the incoming object with the identity "Foo" should replace what still is "Bar" in persistent storage. Similar to changing your name: new passport, new contracts, new anything, but nothing trivial.

#12 - 2011-06-17 17:53 - Bastian Waidelich

Karsten Dambekalns wrote:

> *[...] the title is part of the identity, and changing this is non-trivial at best.*

I see, but do you have a idea/suggestion on how to solve this? Or do we have to revert the models to use a uuid column?

#13 - 2011-08-04 08:50 - Sebastian Kurfuerst

- Target version set to 1.0 beta 1

#14 - 2011-08-24 08:15 - Karsten Dambekalns

With the new persistence behavior this should be solved, check!

#15 - 2011-08-26 18:42 - Karsten Dambekalns

- Status changed from Needs Feedback to Accepted

- Target version changed from 1.0 beta 1 to 1.0 beta 2

I am rather convinced this works as expected now, but I'll double check next week.

#16 - 2011-09-07 17:03 - Karsten Dambekalns

- Status changed from Accepted to Resolved

- % Done changed from 0 to 100

The way used in https://review.typo3.org/#change,4692 is correct and then updating works fine. This can probably be made easier, but that is handled in separate issues.