TYPO3.Flow - Bug #47975

initializeObject in a Entity is called at a time where no properties are loaded

2013-05-06 11:52 - Benno Weinzierl

Status:	Resolved	**Start date:**	2013-05-06
Priority:	Must have	**Due date:**	
Assignee:	Robert Lemke	**% Done:**	0%
Category:	Object	**Estimated time:**	0.00 hour
Target version:	2.0.1		
PHP Version:	5.4	**Complexity:**	
Has patch:	No		

Description

I have an Entity which is loaded during validation of a Action Argument. (The Action Argument is another Entity with relations to the Entity in question)

When initializeObject() is called all properties are missing and have not been loaded from persistence.

Here is the Backtrace:
https://gist.github.com/anonymous/5524256

Related issues:

Is duplicate of TYPO3.Flow - Bug #43659: Proxy class building calls __constru...	**Resolved**	**2012-12-06**

History

#1 - 2013-05-06 17:29 - Alexander Berl

The real problem is, that the Doctrine Proxy __load() calls __wakeup() before the properties are actually loaded.

Here's a pseudo call stack for the class hierarchy of the proxies to original class, when the doctrine proxy is the entry point:

```
Class hierarchy:    Doctrine Proxy (DP) > Flow Proxy (FP) > Original Class (OC)

Call stack1: Doctrine Proxy gets loaded due to property access or via the GenericObjectValidator
DP->__load()
    DP->__wakeup()
        DP->__load()
        FP->__wakeup()
            OC->__wakeup()
            DP->initializeObject(2)
                DP->__load()
                OC->initializeObject()
DP->_entityPersister->load($this->_identifier, $this)

Call stack2: Doctrine Proxy gets unserialized (e.g. unserialize('O:...') hacks used both in Flow and Doctrine)
DP->__wakeup()
    DP->__load() [I think this is not really executed, since $this->_entityPersister would be empty in an empt
y unserialization]
    FP->__wakeup()
        OC->__wakeup()
        DP->initializeObject(2)
            DP->__load()
            OC->initializeObject()
```

Indentation refers to the call nesting level. As you can see, in both cases the initializeObject method gets called before the object properties are loaded. However, I'm not sure how to best deal with the premature __wakeup() in __load() and preventing empty unserialization wakeups from triggering initializeObject.

A possible check could be that if __isInitialized__ is set and true and __entityPersister is set, or if __isInitialized__ is set and false then skip the initializeObject invokation in the FlowProxy. You'd completely loose the initializeObject call in those cases though and at least in the first this is not desired I guess.

#2 - 2013-05-09 18:55 - rottenrice no-lastname-given

I have the same problem but my call stack differs from your call stack.

My two Entities:

```
/** @Flow\Entity */
class Foo {
 public function __construct() {
  echo '__construct()';
 }
 public function initializeObject($arg) {
  echo 'initializeObject('.$arg.')';
 }
 public function injectSettings() {
  echo 'injectSettings()';
 }
}

/** @Flow\Entity */
class Bar extends Foo {
 public function __construct() {
  echo '__construct()';
  parent::__construct();
 }
}
```

Example code:

```
$bar = new Bar();
```

Output:

```
__construct()
__construct()
initializeObject(1)
injectSettings()
initializeObject(1)
```

Reason:
__construct() of Flow-Proxy (Foo)

```
[...]

if ('Foo' === get_class($this)) {
 $this->Flow_Proxy_injectProperties();
}
$this->initializeObject(1);
```

The injectProperties() in not executed but the initializeObject() is executed :/

#3 - 2013-05-14 17:47 - Adrian Föder

I also had that; and I realized it's because of this gem here:

```
\Doctrine\ORM\Mapping\ClassMetadataInfo::newInstance

    public function newInstance()
    {
        if ($this->_prototype === null) {
            $this->_prototype = unserialize(sprintf('O:%d:"%s":0:{}', strlen($this->name), $this->name));
        }

        return clone $this->_prototype;
    }
```

This is a dummy way to get an object without the constructor being called; but unfortunately this calls __wakeup and hence initializeObject(), of course without any properties then.

At the end, I had initializeObject() being called twice, once in that erroneous case and once when it's actually intended.

#4 - 2013-05-21 11:31 - Robert Lemke

- Status changed from New to Accepted

Needs to be verified and solved for 2.0

#5 - 2013-08-14 15:35 - Karsten Dambekalns

- Target version changed from 2.0 to 2.0.1

#6 - 2013-10-11 10:10 - Robert Lemke

- Category changed from Persistence to Object

- Assignee set to Robert Lemke

- PHP Version set to 5.4

#7 - 2013-10-11 10:12 - Robert Lemke

I think that this is a duplicate of #43659

#8 - 2013-10-11 10:13 - Robert Lemke

- Status changed from Accepted to Closed

Let's see if the fix for #43659 helps

#9 - 2013-10-11 11:03 - Andreas Förthner

- Status changed from Closed to New

The problem still exists, opening the issue again...

#10 - 2014-07-11 20:19 - Robert Lemke

- Status changed from New to Resolved

Applied in changeset 6d2ea6b3fc5db90f3c7e6ccd0dea5304ec32ef5e.