TYPO3.Flow - Bug # 29449

Status:	Resolved	**Priority:**	Must have
Author:	Sebastian Kurfuerst	**Category:**	Reflection
Created:	2011-09-02	**Assigned To:**	Sebastian Kurfuerst
Updated:	2011-09-05	**Due date:**	
PHP Version:			
Has patch:			
Complexity:			
Affected Flow version:			
Subject:	Add safeguard preventing to reflect Doctrine Proxies		

Description

In some custom application of mine, I have accidentally reflected a doctrine proxy class; which lead to the ReflectionService being called like this:

```
ReflectionService::reflectClass('TYPO3\FLOW3\Persistence\Doctrine\Proxies\SandstormMediaWorkflowDomainModelProcesInstanceProxy')
```

The ReflectionService reflected the class, returned the expected result and then **saved** the information about the above classname in the cache.

Unluckily, the above class implements multiple interfaces:

- Doctrine\ORM\Proxy\Proxy
- TYPO3\FLOW3\AOP\ProxyInterface (as also some aspects apply to the above class)

Now, on the next FLOW3 run, the following happens:

- The Object\Configuration\ConfigurationBuilder iterates through all available class names, thus it will also iterate over Doctrine\ORM\Proxy\Proxy
- The ConfigurationBuilder then asks ReflectionService::getDefaultImplementationClassNameForInterface('Doctrine\ORM\Proxy\Proxy')
- The ReflectionService returns **our above initial class (....Proxies\SandstormMedia.....Proxy)**; because that's the only known implementation of the Proxy interface
- then, various weird things can happen... for example ReflectionService::getMethodTagsValues(...) might fail because the ...Proxy class could not be loaded by the autoloader.

This leads to errors like:

```
Uncaught Exception in FLOW3 Class TYPO3\FLOW3\Persistence\Doctrine\Proxies\SandstormMediaWorkflowDomainModelProcessInstanceProxy does not exist thrown in file Packages/Framework/TYPO3.FLOW3/Classes/Reflection/ReflectionService.php in line 595 #0 TYPO3\FLOW3\Error\DebugExceptionHandler::echoExceptionCli() Packages/Framework/TYPO3.FLOW3/Classes/Error/AbstractExceptionHandler.php:60 #1 TYPO3\FLOW3\Error\AbstractExceptionHandler::handleException()
```

So, the behavior is as follows:

- on first hit, everything works.
- on second hit, **at compile time**, the system seems to break randomly (took me quite some time to figure out what happens)

Root Cause

The issue was caused by some custom package of mine, calling the ReflectionService in a wrong/unspecified way. I did
$reflectionService->getClassSchema(get_class($object));
where $object was a doctrine proxy. And this lead to the above behavior.

Possible Solutions

We somehow need to prevent the above root cause. For that, I see three possibilities, all of them adding a check to ReflectionService::reflectClass()

1. fail silently when detecting that the incoming class name is a doctrine proxy
2. throw an exception when detecting that the incoming class name is a doctrine proxy
3. when detecting a doctrine proxy, find out the class name the user **really** wanted to reflect instead, and return this reflection data instead.

3) might have some side-effects; as we would need to touch most of the methods in the ReflectionService, everywhere $this->reflectClass() is called. That's **why my preferred solution is 2)**.

How to continue

I'll fix this issue; I'd just like your opinion which solution you like most.

Greets, Sebastian

Associated revisions

Revision 98f877bb - 2011-09-02 08:31 - Sebastian Kurfuerst

[BUGFIX] (Reflection): Prevent use of ReflectionService for Doctrine Proxies

When calling the ReflectionService with a class name for a doctrine proxy,
really weird side-effects can happen, as the Doctrine Proxy class is then
also stored in the Reflection Cache. On the next compilation run, really
weird side-effects can happen. See the corresponding issue for a full
description.

By throwing an exception when a doctrine class is reflected, we prevent
the issue from appearing in the first place.

Resolves: #29449
Change-Id: Ia709b70e4e31facfd88563c5836009d7cee6d7b2

History

#1 - 2011-09-02 07:31 - Robert Lemke

> *We somehow need to prevent the above root cause. For that, I see three possibilities, all of them adding a check to ReflectionService::reflectClass()*
>
> *1. fail silently when detecting that the incoming class name is a doctrine proxy*
> *2. throw an exception when detecting that the incoming class name is a doctrine proxy*
> *3. when detecting a doctrine proxy, find out the class name the user **really** wanted to reflect instead, and return this reflection data instead.*

I also opt for 2). The question is if the Doctrine Proxy should be or can be identified by an interface which is part of the FLOW3 package. But that's probably of minor importance.

#2 - 2011-09-02 07:50 - Karsten Dambekalns

- Status changed from New to Accepted

Definitely solution 2)

#3 - 2011-09-02 07:52 - Karsten Dambekalns

Robert Lemke wrote:

> *The question is if the Doctrine Proxy should be or can be identified by an interface which is part of the FLOW3 package.*

Why? There are a number of places where code is tied to Doctrine, and if it is, this should be visible IMHO. Effectively *hiding* that fact in some other interface name doesn't help, does it?

#4 - 2011-09-02 08:31 - Mr. Hudson

- Status changed from Accepted to Under Review

Patch set 1 of change Ia709b70e4e31facfd88563c5836009d7cee6d7b2 has been pushed to the review server.
It is available at http://review.typo3.org/4738

#5 - 2011-09-05 12:36 - Sebastian Kurfuerst

- Status changed from Under Review to Resolved
- % Done changed from 0 to 100

Applied in changeset commit:98f877bb3524bff1db2c991b494d1cc10682c2ad.