TYPO3.Flow - Bug #52909

Class Loader fallback to non-proxy hides fatal errors

2013-10-17 11:24 - Stephan Schuler

Status:	New	**Start date:**	2013-10-17
Priority:	Should have	**Due date:**	
Assignee:		**% Done:**	0%
Category:	Core	**Estimated time:**	0.00 hour
Target version:			
PHP Version:		**Complexity:**	
Has patch:	No		

Description

Hey there.

This commit is slightly related to #46689. But since #46689 has drifted to focusing doc parsing and annotation, I will explain this here instead of #46689.

The current strategy is:

- ClassLoader uses a PhpFrontend which uses a SimpleFileBackend to cache proxy class files.
- The SimpleFileBackend catches exceptions and returns FALSE if any exception or notice is thrown inside of an included file.

This leads to falling back to the very orignial file inside of the package directory whenever a notice accures inside of a proxy class file.

Having a non proxied original file when you expect the proxied one leads to kind of unexplainable misbeavior.

Real world example:

Think about a class having a method that has a type hint, e.g. *RsaWalletServicePhp::decryptWithPrivateKey($cipher, \TYPO3\Flow\Security\Cryptography\OpenSslRsaKey $privateKey)*.

Now extend this class and overwrite it without the type hint: *ExtendedRsaWalletServicePhp::decryptWithPrivateKey($cipher, $privateKey)*.

The proxy class file gets created and exists in the proxy class files directory.
The class loader tries to include it through te SimpleFileBackend and runs into the following exception:

```
Runtime Notice: Declaration of ExtendedRsaWalletServicePhp::decryptWithPrivateKey() should be comp
atible with that of TYPO3\Flow\Security\Cryptography\RsaWalletServicePhp_Original::decryptWithPriv
ateKey() [...]
```

Now the fallback strategy of the ClassLoader kicks in and includes the original file that exists in the package directory.

This obviously results in confusing behavior, since the class itself exists in the PHP process environment, but it's not the proxied one so it's not initialized by the AOP stuff. The example here has no settings, which means it doesn't know where the RSA key file can be found, which means no keys available.

How to notice this and how it confuses

The thing is: There is no exception, so there is no log file in the Logs/Exceptions directory. And there is no log line in the Logs/System_*.log file.

Thinking about the above example, there are two relevant proxy class files. One holds the ExtendedRsaWalletServicePhp and ExtendedRsaWalletServicePhp_Original classes, the other one holds the RsaWalletServicePhp and RsaWalletServicePhp_Original classes.

You would expect to have a valid ExtendedRsaWalletServicePhp object that is fully loaded by AOP, but it isn't.

I added breakpoints to both, the proxied ExtendedRsaWalletServicePhp:__construct and the proxied

RsaWalletServicePhp:__construct.

The very confusing thing:

- The proxied ExtendedRsaWalletServicePhp:__construct isn't executed (obviously, because it's not the proxy class which gets included but the original package file.
- The proxied RsaWalletServicePhp:__construct is executed, but get_class($this) shows that it's a ExtendedRsaWalletServicePhp object.
- Now you start to thing: Why is the __construct of the class skipped in this situation? Points to a PHP bug (which it isn't, of course).

Fix:

The ClassLoder should know if there has to be a proxy class file or not. Now the ClassLoader should throw an exception if the Reflection knows about proxy class files but the PhpBackend returned FALSE and indicates the "did not load" situation.

It's related to this commit:
https://git.typo3.org/Packages/TYPO3.Flow.git/commitdiff/f04343e10cef18e3e76bb302b840c1313225b57e

The real problem is: The class loader doesn't take care of the result, it simply falls back to the package file.

I tried PHP versions 5.3.8, 5.3.10, 5.3.24 and a 5.4.x I don't remember exactly.

Regards,
Stephan.