TYPO3.Flow - Bug # 46424

Status:	Resolved	**Priority:**	Must have
Author:	Adrian Föder	**Category:**	Object
Created:	2013-03-19	**Assigned To:**	Robert Lemke
Updated:	2013-04-26	**Due date:**	
PHP Version:			
Has patch:	No		
Complexity:			
Affected Flow version:	Git master		
Subject:	Infinite recursive call in DependencyProxy		

Description

I have the following situation,

[19-Mar-2013 09:02:22 UTC] PHP Fatal error: Maximum function nesting level of '500' reached, aborting! in ...\TYPO3.Flow\Classes\TYPO3\Flow\Object\ObjectManager.php on line 160
[19-Mar-2013 09:02:22 UTC] PHP Stack trace:
[19-Mar-2013 09:02:22 UTC] PHP 1. {main}() C:\Users\afoeder\PhpstormProjects\acme\Distribution\Web\index.php:0
[19-Mar-2013 09:02:22 UTC] PHP 2. TYPO3\Flow\Core\Bootstrap->run()
C:\Users\afoeder\PhpstormProjects\acme\Distribution\Web\index.php:27
[19-Mar-2013 09:02:22 UTC] PHP 3. TYPO3\Flow\Http\RequestHandler->handleRequest()
...\TYPO3.Flow\Classes\TYPO3\Flow\Core\Bootstrap.php:113
[19-Mar-2013 09:02:22 UTC] PHP 4. TYPO3\Flow\Object\DependencyInjection\DependencyProxy->setRequest()
...\TYPO3.Flow\Classes\TYPO3\Flow\Http\RequestHandler.php:122
[19-Mar-2013 09:02:22 UTC] PHP 5. TYPO3\Flow\Object\DependencyInjection\DependencyProxy->__call()
...\TYPO3.Flow\Classes\TYPO3\Flow\Http\RequestHandler.php:122
[19-Mar-2013 09:02:22 UTC] PHP 6. call_user_func_array()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:96
[19-Mar-2013 09:02:22 UTC] PHP 7. TYPO3\Flow\Object\DependencyInjection\DependencyProxy->__call()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:0
[19-Mar-2013 09:02:22 UTC] PHP 8. call_user_func_array()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:96
[19-Mar-2013 09:02:22 UTC] PHP 9. TYPO3\Flow\Object\DependencyInjection\DependencyProxy->__call()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:0
[19-Mar-2013 09:02:22 UTC] PHP 10. call_user_func_array()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:96
[19-Mar-2013 09:02:22 UTC] PHP 11. TYPO3\Flow\Object\DependencyInjection\DependencyProxy->__call()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:0
.
.
.
.
[19-Mar-2013 09:02:23 UTC] PHP 492. call_user_func_array()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:96
[19-Mar-2013 09:02:23 UTC] PHP 493. TYPO3\Flow\Object\DependencyInjection\DependencyProxy->__call()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:0
[19-Mar-2013 09:02:23 UTC] PHP 494. call_user_func_array()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:96
[19-Mar-2013 09:02:23 UTC] PHP 495. TYPO3\Flow\Object\DependencyInjection\DependencyProxy->__call()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:0
[19-Mar-2013 09:02:23 UTC] PHP 496. TYPO3\Flow\Object\DependencyInjection\DependencyProxy->_activateDependency()

...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:95
[19-Mar-2013 09:02:23 UTC] PHP 497. Closure->__invoke()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:58
[19-Mar-2013 09:02:23 UTC] PHP 498.
TYPO3\Flow\Security\Aspect\PersistenceQueryRewritingAspect->TYPO3\Flow\Security\Aspect\{closure}()
...\TYPO3.Flow\Classes\TYPO3\Flow\Object\DependencyInjection\DependencyProxy.php:58
[19-Mar-2013 09:02:23 UTC] PHP 499. TYPO3\Flow\Object\ObjectManager->get()
C:\Users\afoeder\PhpstormProjects\acme\Distribution\Data\Temporary\Development\Cache\Code\Flow_Object_Classes\TYPO3_Flow_Se
3_Flow_Security_Aspect_PersistenceQueryRewritingAspect.php:603

When debugging the DependencyProxy's __call() method, it turns out that it happens on TYPO3\Flow\Security\Context's setRequest method.

It looks like this especially happens when I become logged out on a "protected" page, maybe because of session timeout or such.

Related issues:

related to TYPO3.Flow - Bug # 46210: securityContext->getParty() in the initi...	**Needs Feedback**	**2013-03-12**

Associated revisions

Revision a532ede6 - 2013-04-24 18:00 - Robert Lemke

[BUGFIX] Lazy DI causes endless loop for certain session objects

Fixes an issue with the Lazy Dependency Injection mechanism which caused
and endless loop if session-scoped objects were unserialized and later on
injected lazily.

Background:

when a session is resumed (rather early in the HTTP Request Handler), the
objects contained in the session are unserialized and register their
instance automatically at the Object Manager. If
such an object, for example the Security Context, is later on injected
lazily into another class, the generated proxy code will overwrite the
instance which was previously set at the Object Manager.

In the reported case the RequestHandler retrieved the Security Context
via ObjectManager->get() and received a DependencyProxy instead of the
expected real instance. On using it, the DependencyProxy called the
ObjectManager->get() method to retrieve the real instance which resulted
in a recursion.

This patch corrects the generated proxy code to check for an existing
real instance before trying to generate a Dependency Proxy.

Change-Id: I3b8997ce07a8d36e7cc6b47e31ea8d52547d10a1
Resolves: #46424
Releases: master, 2.0

Revision be43db2a - 2013-04-26 12:17 - Robert Lemke

[BUGFIX] Lazy DI causes endless loop for certain session objects

Fixes an issue with the Lazy Dependency Injection mechanism which caused
and endless loop if session-scoped objects were unserialized and later on
injected lazily.

Background:

when a session is resumed (rather early in the HTTP Request Handler), the
objects contained in the session are unserialized and register their
instance automatically at the Object Manager. If
such an object, for example the Security Context, is later on injected
lazily into another class, the generated proxy code will overwrite the
instance which was previously set at the Object Manager.

In the reported case the RequestHandler retrieved the Security Context
via ObjectManager->get() and received a DependencyProxy instead of the
expected real instance. On using it, the DependencyProxy called the
ObjectManager->get() method to retrieve the real instance which resulted
in a recursion.

This patch corrects the generated proxy code to check for an existing
real instance before trying to generate a Dependency Proxy.

Change-Id: I3b8997ce07a8d36e7cc6b47e31ea8d52547d10a1
Resolves: #46424
Releases: master, 2.0

History

#1 - 2013-03-19 10:26 - Robert Lemke

- Status changed from New to Accepted

- Assigned To set to Robert Lemke

#2 - 2013-03-26 17:44 - Adrian Föder

ah ok that's pretty straight-forward: the problem occurs very early,

```
4 TYPO3\Flow\Object\DependencyInjection\DependencyProxy::__call("setRequest", array|1|)
3 TYPO3\Flow\Object\DependencyInjection\DependencyProxy::setRequest(TYPO3\Flow\Mvc\ActionRequest)
2 TYPO3\Flow\Http\RequestHandler::handleRequest()
1 TYPO3\Flow\Core\Bootstrap::run()
```

So actually, it happens in \TYPO3\Flow\Http\RequestHandler::handleRequest, line 122:

```
1$this->securityContext->setRequest($actionRequest);
```

This securityContext property is the DependencyProxy; however, it looks like the "realDependency" is again the DependencyProxy (maybe that even should be blocked with an exception, again).

Just in order to emphasize, this exception is thrown in my case:

```
1  public function _activateDependency() {
2      $realDependency = $this->builder->__invoke();
3      if ($realDependency instanceof $this) {
4          throw new \Exception('The actual dependecy which should be proxies by the Dependency Proxy is again the Dependency Proxy which must never occur.');
5      }
```

#3 - 2013-03-27 09:30 - Adrian Föder

("split" this topic to an additional one, #46716)

#4 - 2013-03-27 12:32 - Adrian Föder

- File 500_Internal_Server_Error.pdf added

I debugged the invocations of ObjectManager->get(...Security\Context), and,

- the first invocation instantiated and returned the correct class
- the second one returned the correct object from the $this->objects stack
- the third invocation also returned from the $this->objects stack, but **the dependency proxy!**

So, the whole HTTP request handling thing already only get the (incorrect) DependencyProxy object; means, \TYPO3\Flow\Http\RequestHandler::resolveDependencies and \TYPO3\Flow\Http\RequestHandler::handleRequest all has only the DependencyProxy (see Note 2 here).

The last ObjectManager->get(...Security\Context) invocation which returns the **correct** SecurityContext is one related to AspectQueryRewriting, Content Security etc...
See attached file for a backtrace (I made a dummy exception).

#5 - 2013-04-23 18:51 - Gerrit Code Review

- Status changed from Accepted to Under Review

Patch set 2 for branch **master** has been pushed to the review server.
It is available at https://review.typo3.org/20113

#6 - 2013-04-24 18:00 - Gerrit Code Review

Patch set 3 for branch **master** has been pushed to the review server.
It is available at https://review.typo3.org/20113

#7 - 2013-04-24 18:02 - Gerrit Code Review

Patch set 1 for branch **2.0** has been pushed to the review server.
It is available at https://review.typo3.org/20136

#8 - 2013-04-24 18:22 - Adrian Föder

- Status changed from Under Review to Resolved

#9 - 2013-04-25 09:38 - Gerrit Code Review

- Status changed from Resolved to Under Review

Patch set 2 for branch **2.0** has been pushed to the review server.
It is available at https://review.typo3.org/20136

#10 - 2013-04-26 11:33 - Adrian Föder

- Status changed from Under Review to Resolved

#11 - 2013-04-26 12:17 - Gerrit Code Review

- Status changed from Resolved to Under Review

Patch set 3 for branch **2.0** has been pushed to the review server.
It is available at https://review.typo3.org/20136

#12 - 2013-04-26 12:35 - Anonymous

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

Applied in changeset commit:be43db2a12c63d1da71272f5310186d56dceaa7b.

Files

500_Internal_Server_Error.pdf	58 kB	2013-03-27	Adrian Föder