TYPO3 Flow Base Distribution - Story # 11169

Status:	Resolved	**Priority:**	Should have
Author:	Robert Lemke	**Category:**	
Created:	2010-11-30	**Assigned To:**	Robert Lemke
Updated:	2011-05-06	**Due date:**	
Subject:	Implement new operator support		

Description

Refactor FLOW3 so that the PHP "new" operator can be used instead of ObjectManager->create()

Subtasks:

Task # 13703: Fix file monitoring / optimize compile step	**Resolved**
Task # 13702: Replace remaining uses of objectManager->create() with new and adjust tes...	**Resolved**
Task # 12099: Check feasibility and create rough concept	**Resolved**
Task # 12100: Implement a generic Proxy Builder	**Resolved**
Task # 13604: Fix bugs and adjust affected code	**Resolved**

History

#1 - 2010-11-30 20:33 - Benjamin Eberlei

I want to throw in https://github.com/sebastianbergmann/php-test-helpers

This is called "test-helpers", however it is exactly useful for overwriting the new operator to implement AOP.

#2 - 2010-12-16 19:36 - Sebastian Kurfuerst

Hey,

as already discussed with you @Robert, I don't see this as a high priority right now. Just wanted to add this to the issue, so it does not get lost :-)

Greets,
Sebastian
PS: @Benjamin: It's great that you're active here, helping us with the Doctrine evaluation :-)

#3 - 2010-12-21 16:24 - Robert Lemke

An important argument against using the new operator is testability. If classes create their own prototypes (prototypes in the context of FLOW3), there is no easy way to mock them. That's exactly the reason why PHP-test-helpers were developed.

On the other hand, if we can afford to require the use of such a custom PHP extension for testing then it might make sense to keep on exploring this topic.

#4 - 2010-12-21 16:33 - Benjamin Eberlei

Hm I disagree!

You already mention prototypes in the context of Flwo3and mix it with php test helpers. Your reasoning would imply that php-test-helpers was developed for Flow3. It was developed for applications that don't use dependency injection.

But Dependency Injection is independent of the problem newable vs injectable, which is discussed here:

http://misko.hevery.com/2008/09/30/to-new-or-not-to-new/

and also in Evans Domain Driven Design (Entities and Services chapter).

I don't say you ObjectManager::create is bad per se. Its just bad for "newables" (Entities, Value objects and such).

#5 - 2010-12-21 16:58 - Robert Lemke

Benjamin Eberlei wrote:

> *You already mention prototypes in the context of Flwo3and mix it with php test helpers. Your reasoning would imply that php-test-helpers was developed for Flow3. It was developed for applications that don't use dependency injection.*
>
> *But Dependency Injection is independent of the problem newable vs injectable, which is discussed here:*
>
> *http://misko.hevery.com/2008/09/30/to-new-or-not-to-new/*

Misko argues that there's no need to mock "leafs of the application" because they are more or less like strings. I don't agree with that. I can easily imagine Entities which rely on a complex set of other objects which I'd rather unit-test separately and not alltogether. If I test a "Conference" object I want to see my test pass or fail because "Conference" is implemented right or wrong. It's confusing to see "Conference" tests fail because the "Address" object used by the "Venue" object created by the "Conference" object is buggy.

> *and also in Evans Domain Driven Design (Entities and Services chapter).*
> *I don't say you ObjectManager::create is bad per se. Its just bad for "newables" (Entities, Value objects and such).*

Hmm, I don't know. I guess you can produce okay code either way, it's rather a question of style. What I find appealing with the new-operator approach is that it a) doesn't require an Object Manager and b) is less to type. However, the ObjectManager->create() approach is easier to mock and more explicit about who's responsible for object creation.

Certainly, we could go for a mixture of both (like Symfony tends to do), but that's the option I dislike most. I'd rather have everything consistent following the same conventions.

#6 - 2010-12-21 17:27 - Benjamin Eberlei

Miskos main argument is not about the leafes of an application, its about hard to test code if you violate the injectable vs newable rule, because in this case to be able to test newables you might need injectables. This is cumbersome to setup. In the case of FLOW3 its easy to explain:

Why do i need to configure tons of mocking lines just to be able to test my simple "BlogPost" entity. Its supposed to be a domain object, not knowing anything about technical blabla such as an object manager. There is a a considerable difference between:

```
class BlogPost
{
    public function addComment($message, $email)
    {
        $comment = $this->objectManager->create("Comment", array($message, $email));
        $this->comments[] = $comment;
        return $comment;
```

```
    }
}

class BlogPost
{
    public function addComment($message, $email)
    {
        $comment = new Comment($message, $email);
        $this->comments[] = $comment;
        return $comment;
    }
}
```

in regards to testing:

```
class BlogPostTest
{
    public function testSimpleNewable()
    {
        $post = new BlogPost();
        $comment = $post->addComment("hello world!", "kontakt@beberlei.de");

        $this->assertType("Comment", $comment);
        $this->assertEquals("hello world!", $comment->getMessage());
        $this->assertEquals(1, count($post->getcomments());
    }

    public function testComplexObjectManagerTest()
    {
        $message = "hello world!";
        $email = "kontakt@beberlei.de";

        $objectManager = $this->getMock('Some/Very/Long/PathTo/ObjectManager');
        $objectManager->expects($this->at(0));
                ->method('create');
                ->with($this->equalTo('Comment), $this->equalTo(array($message, $email))
                ->will($this->returnValue(new Comment($message, $email));

        $post = new BlogPost();
        $post->setObjectManager($objectManager); // rather simple, but what if you rely on reflection only here in your model?
        $comment = $post->addComment("hello world!", "kontakt@beberlei.de");

        $this->assertType("Comment", $comment);
        $this->assertEquals("hello world!", $comment->getMessage());
        $this->assertEquals(1, count($post->getcomments());
    }
}
```

This is miles apart in terms of simplicity. Because test two violates the rule, never depend on injectables in your newables.

Please also review the section on factories and aggregate roots in DDD, it specificially says entities / aggregate roots can create objects themselves

not some object manager.

Additionally the current approach means that you need the object manager as reference in EVERY object. Becaues whenever i want to create a new object I also need an object manager. That all just leads to very complicated and repetitive code. (as shown in the above test-example).

#7 - 2010-12-21 18:41 - Robert Lemke

- Assigned To set to Robert Lemke

#8 - 2010-12-21 18:41 - Robert Lemke

- Status changed from New to On Hold

#9 - 2011-01-06 14:13 - Sebastian Kurfuerst

- Status changed from On Hold to New
- Parent task changed from #10273 to #11946

#10 - 2011-01-06 14:13 - Sebastian Kurfuerst

- Tracker changed from Task to Story
- Start date set to 2011-01-06
- Parent task deleted (#11946)

#11 - 2011-01-06 14:22 - Sebastian Kurfuerst

- Position set to 2

#12 - 2011-01-07 12:34 - Robert Lemke

- Status changed from New to Accepted

#13 - 2011-01-11 15:19 - Robert Lemke

- Subject changed from Explore possibility to use new operator instead of ObjectManager->create() to Implement new operator support

#14 - 2011-01-11 15:25 - Benjamin Eberlei

So how will this work? Your tweet about it didn't release any technical details :-)

#15 - 2011-05-05 13:51 - Robert Lemke

- Project changed from Core Team to Base Distribution
- Target version deleted (788)

#16 - 2011-05-06 08:12 - Robert Lemke

- Project changed from Base Distribution to TYPO3 Flow Base Distribution

#17 - 2011-05-06 08:12 - Robert Lemke

- Target version set to 1.0 beta 1
- Position deleted (17900)
- Position set to 1

#18 - 2011-05-06 08:14 - Robert Lemke

- Project changed from TYPO3 Flow Base Distribution to Base Distribution
- Target version changed from 1.0 beta 1 to 1228
- Position deleted (2)
- Position set to 1

#19 - 2011-05-06 08:17 - Robert Lemke

- Project changed from Base Distribution to TYPO3 Flow Base Distribution

- Target version deleted (1228)

#20 - 2011-05-06 08:17 - Robert Lemke

- Target version set to 1.0 beta 1

- Position deleted (1)

- Position set to 1

#21 - 2011-05-06 09:24 - Robert Lemke

- Status changed from Accepted to Resolved