Subject: (re-)introducing Dependency Injection
Posted by DannyBoyPoker on Sat, 06 Apr 2013 03:03:46 GMT
View Forum Message <> Reply to Message

I've seen this:
http://www.tonymarston.net/php-mysql/dependency-injection-is -evil.html

What does "Dependency" mean? etc. And: 'Using terms such as "dependency", "coupling" and "cohesion" can be confusing unless you identify exactly what they mean.'

Here I learn, that 'coupling' describes how modules interact. Lower coupling is better. And:
--
Tightly coupled systems tend to exhibit the following developmental characteristics, which are often seen as disadvantages:
A change in one module usually forces a ripple effect of changes in other modules. Assembly of modules might require more effort and/or time due to the increased inter-module dependency.
A particular module might be harder to reuse and/or test because dependent modules must be included.
--

And, I learn of 'too many dunderheads out there', etc. 'And these people call me crazy!' And, I've been there

You quote an article, that has this line: 'Consider the below example we have a customer class which contains an address class object etc.'

And, your impatience w/this is pretty boundless. So, of this: 'Customer class is aware of the address class type. So if we add new address types like home address, office address it will lead to changes in the customer class also as customer class is exposed to the actual address implementation'.

You offer here, that: This is a meaningless accusation which describes a non-problem. It is a simple fact that the customer object must communicate with the address object in order to obtain the customer's address, so the customer object *MUST* know that the address object exists, and it *MUST* know which method to use to get the data it wants. It is simply not possible to write code which effectively says "get me some data from an unknown object using an unknown method" - unless, of course, you live in cloud cuckoo land.

I don't gather that the author of this article lives in cloud cuckoo land.

It may be a simple fact that etc., as you say, but this is not a meaningless accusation that describes a non-problem. I suppose that it actually is a non-problem if it can't be solved. Then, is it possible to write code which effectively says "get me some data from an unknown object using an unknown method"? Given that he lists various types of DI to implement what he wants (specifically, he mentions four), and how 'we can actually implement these', and here, he talks about using factory, and about a container, I'll leave this aside.

The point I want to get to, is, are you addressing, here, his two points? Which are, first, that main

classes aggregating other classes should not depend on the direct implementation of the aggregated classes. What then? Well, both the classes should, putatively, depend on abstraction. And, second, abstraction should not depend on details (rather, details should depend on abstraction). This is what he calls tight coupling, which is what he calls the problem.

And, you ask: 'all the "problems" identified in that article do not exist in my application, so if the problems do not exist then can you please explain to me in words on one syllable what benefits I would obtain by implementing this solution?'

You also cite an article:
http://ralphschindler.com/2011/05/18/learning-about-dependen cy-injection-and-php

And you say: 'I therefore consider the article to be a waste of space and unworthy of serious consideration, especially from an OO heretic such as me.'

Okay, so, um, okay, I'm not picking on you--this is a cry for help. You asked. I note, that this article says: 'This article is not about the intricacies and implementation details of DI containers and DI frameworks.'

I kind of gather a comportment from you on this, maybe it's possible to caricature, that you're saying: Decouple A from B? Why? A needs B!

To this general point, I have a general answer, that therein lies the problem. If I have a plug, and then I have an outlet, what is the reasonable response? However, software systems, so I am told, become unmaintainable because we let them grow on their own, without keeping a keen eye on how that growth is proceeding, until, one day, we realize that our software is a tangled mess of copied code and interwoven dependencies. And we have no one to blame but ourselves. Which I write, for my own benefit, I'm experienced, with taking action, without considering the reasons or implications.

More specifically, I see this: http://www.radicore.org/whatisradicore.php

8. Developers are spared the chore of designing and coding their hierarchy of menus as RADICORE comes supplied with a pre-built MENU system which allows menu pages to be constructed and maintained using a standard set of online screens.

9. Developers are spared the chore of designing and building a security system as RADICORE comes supplied with a pre-built RBAC system which allows access control lists to be constructed and maintained using a standard set of online screens.

However, I notice, that, in fact, the 'menu' subsystem, is really a menu and security system. Instead of there being a separate, global security system, security is handled by the menu subsystem. "Menu", in fact, maintains a list of the subsystems, and has a pointer to itself, as one of them. The users, and roles, tasks, also, are maintained, by the menu subsystem. Nothing runs w/out the menu subsystem, then. You can't not have a menu.

It occurs to me, that one might want to swap out the menu system, for another. This wouldn't be a simple as dropping in another menu subsystem, as a replacement. You can't delete the

menu subsystem. I might have supposed, that menu systems, and, more generally, structure & navigation,
would be areas in which one might want to mix and match various components. One might have a few menu
components, and display them both on a page (which is more the rule, than the exception).

One might, similarly, take this view of access & security. Furthermore, these are maybe two different things. Authentication would be a matter of site access, for example. One might describe password management, backup, site monitoring, as security.

How to accomplish this?
