What's your opinion on MS CAB (Composite Application Block)?

Asked 15 years, 11 months ago Modified 14 years, 4 months ago

Viewed 4k times

6

I'm in the process of evaluating the use of CAB for a new .net 3.5 winform project

I plan to use the Infragistics toolset, which is known to be 'CAB compliant'

While CAB has the immediate upside of letting me focus on my business instead of coding basic docking/login/etc code, I feel like I would be able to achieve the same level of functionnality quite radpidly by myself (with the added flexibility/reactivity bonus you have when you 'own' the code).

I'm seeking some feedback on Microsoft's CAB from people using it :

1. Have you experienced problems/bugs ?
2. Do you feel like CAB saved your time ?
3. Are there extra functionalities I don't know about (beside Docking/Login/WorkerThreads best practices?)

.net winforms frameworks infragistics cab

Share

Improve this question

Follow

asked Jan 25, 2009 at 14:11

Brann

32.3k ● 33 ● 120 ● 164

The link above is no longer valid. I searched for Composite Application Block and found this Overview of the NetAdvantage CAB Extensibility Components which looks like a good starting point. – surfmuggle Sep 2, 2013 at 20:00

5 Answers

Sorted by: Highest score (default)

7

I had some experience using CAB a couple of years ago and my conclusion was that it too complex and had a steep learning curve. As such the benefits it offered just weren't worth the price of getting up to speed with it. However don't take my word for it, try following some of their labs and see what you think.

Jeremy Miller wrote an excellent series of blog posts about building your own CAB

http://codebetter.com/blogs/jeremy.miller/archive/2007/07/25/the-build-your-own-cab-series-table-of-contents.aspx

these are worth a look as you could just take from there what you need.

My advice would be to get on with your project rather than build a framework up front. As the project develops you should spot opportunities to refactor code into base classes and effectively harvest a framework from your application.

That way you will end up with a framework that meets your needs, and that everyone on the development team will understand. Whatever you do don't build a framework up front - there lies the path to ruin :-)

Share Improve this answer

Follow

answered Jan 27, 2009 at 9:30

gilles27

2,241 ● 6 ● 23 ● 37

4

We have used CAB+SCSF for a couple of projects. The learning curve is indeed steep. You will probably be up to speed after the first month. Other cons:

1. Too much complexity
2. Pattern-itis
3. Code generation bloat
4. Hard to debug

The pros:

Follows the best practices architectural design patterns in the industry:

1. Model-View-Presenter

2. UI Composition
3. Dependency injection, Inversion of control
4. Loosely Coupled Events
5. Modularity
6. etc...

Using CAB-SCSF in the long run will mean less bugs and more maintainable code. If your project can afford the initial hit of the learning curve I will definitely recommend it.

Share Improve this answer Follow

edited Nov 20, 2009 at 0:09

answered Aug 1, 2009 at 5:04

AlejandroR

5,181 ● 4 ● 36 ● 46

2 Point 3 probably should say "Dependency Injection" (or "Inversion of Control Container") – galaktor Aug 21, 2009 at 17:55

1 Learning curve is only steep for those that don't have knowledge of OO Design Patterns. I had a team of 6; 3 knew OO and design patterns, and the other 3 did not. Those that had the knowledge took to CAB/SCSF like ducks to water with immediate productivity increases. Those that didn't, practically drowned; primarily because they cared nothing for OO or design patterns. – Travis Heseman Sep 24, 2009 at 20:30

Loosely Coupled Events can also be a downside... For example, there is no compile-time check that the event

handler signature is as expected by the caller. – Massimiliano Nov 24, 2009 at 14:49

1

The CAB is retired in favor of the SCSF. Both the CAB and SCSF offer some value in terms of standardizing rich client development across projects (if you use them that way), but both are very heavyweight.

Share Improve this answer

Follow

answered Aug 1, 2009 at 5:12

JP Alioto

45.1k ● 6 ● 90 ● 112

3 SCSF encapsulates CAB; CAB is not retired. – Travis Heseman Sep 24, 2009 at 20:26

0

While i've never actually used CAB, it ships with the source code so you'd still have the ability to tweak it to suit your exact needs if you need extra flexibility not provided by the Application Block.

Share Improve this answer

Follow

answered Jan 25, 2009 at 14:21

Michael Prewecki

2,092 ● 4 ● 16 ● 28

I would still be forced to reapply my modifications if I want to upgrade to a new release of CAB, for example. – Brann Jan 25, 2009 at 14:24

Very true but I would leverage the efforts of others until my business logic was sound and then think about any plumbing

enhancements by eliminating irritating dependencies...UX is everything but CAB can't be that bad unless you end up tightly coupling your business logic with the UI.
– Michael Prewecki Jan 25, 2009 at 14:32

0

We have build our enterprise application over the CAB framework, but have modified many parts of it to suit our needs.

From that experience, I can tell that CAB is more suitable, when you need a very flexible and modular architecture and there is clear division between development for the ui, business logic and the data layer.

CAB does have its disadvantages of having lot of generated code and sometimes induces developers to use complex mechanisms to achieve simple results (like overuse of the event publication/subscription model, mvp pattern for even the simplest ui, ...)

If you want an introduction to this, there an excellent series by Rich Newman at http://richnewman.wordpress.com/intro-to-cab-toc/

If you only want to make use of Dependency Injection/Inversion Of Control, then Managed Extensibility Framework (http://mef.codeplex.com/), can be a very good, lightweight alternative.

Share Improve this answer

Follow

answered Aug 26, 2010 at 1:48

EndlessSpace