Is it worth using 3-tier architecture for small(ish) applications

Asked 15 years, 10 months ago Modified 15 years, 10 months ago

Viewed 2k times

6

I'm working on a relatively small asp.net web application and am wondering if there is really a need to employ full n-tier architecture. For an idea of size; there are about 20 database tables.

In the past I have used a 2-tier approach where business logic and data access are grouped together into a single class library with was an asp.net web application forming the UI tier and this seemed to work OK.

Is there a threshold size or some rule of thumb where you should employ n-tier?

.net asp.net design-patterns architecture n-tier-architecture

Share

Improve this question

Follow

asked Feb 5, 2009 at 19:34

Matthew Dresser

11.4k ● 11 ● 78 ● 125

5 Answers

Sorted by: Highest score (default)

14

It may be relatively small now but do you think it might grow in the future? Of your course you have to be pragmatic, but if you already have natural separations of concern within your single assembly then it's fairly easy to split that into two or more assemblies. If the classes themselves combine business logic and data access then you have more work on your hands. However, I'd argue that it's almost no more work to write your data access logic in one assembly and your business logic in the other, in parallel if required, than to write it all in one place in the first place.

Share Improve this answer

Follow

edited Feb 5, 2009 at 20:38

David Segonds

84.7k ● 10 ● 49 ● 66

answered Feb 5, 2009 at 19:37

Adam Ralph

29.9k ● 5 ● 61 ● 68

3

If you anticipate growth or change of any kind, it would help to reinforce explicit tiers now. But if this is your own project and you don't mind ponying up the cost of change or tightly integrated code, then just hack away at it like one layer.

It's all about future cost. In the present, just programming as you go is by far the cheapest, but does not respond to

change very well. With the tier planning, there's an upfront cost, but it can handle change and ripping out whole layer implementations better than the first approach can.

So your rule of thumb is governed by who will pay for the change and when.

Share Improve this answer

Follow

answered Feb 5, 2009 at 19:38

Mark Canlas

9,573 ● 5 ● 45 ● 63

3

Yes it is worth it. What you are proposing is hacking a solution together. I can't tell you how many times this has brought me pain and suffering. The other side of that coin is astronaught developers who overbuild their solutions. You want to avoid this as well.

Be simple, build your three layers, I suggest you try out LINQ to SQL, it makes building the DAL a snap, and then you can focus on building a slim UI, and a BLL that will handle the heavy lifting. It won't take you that much longer to do it and it will allow for unit testing later.

Share Improve this answer

Follow

answered Feb 5, 2009 at 20:06

Al Katawazi

7,242 ● 8 ● 27 ● 39

> Is there a threshold size or some rule of thumb

1

> where you should employ n-tier?

Not one that I'm aware of, but I'll throw this out: if there's a chance this web app can grow (or change backends) and/or somebody else may end up maintaining it, I'd consider going with an n-tier approach.

Share Improve this answer

Follow

answered Feb 5, 2009 at 19:46

Cal Jacobson

2,407 ● 1 ● 26 ● 35

1

Just keep it possible. It's probably overkill now but don't write yourself into a situation where it's impossible. Refactoring on a large scale will take you longer than rebuilding it from scratch (and will likely be just that, rebuilding from scratch).

The last project I worked on (came in late) was written in a way that breaking up the tiers was practically impossible. Data logic interwoven with business logic interwoven with presentation logic. In the short term, it was as good as it was ever going to be because fixing it would've taken several months.

It's not too difficult to keep everything separated so like I said earlier, just keep it possible.

Share Improve this answer

Follow

answered Feb 5, 2009 at 20:10

Austin Salonen

50.2k ● 15 ● 111 ● 140