Is data binding a bad idea?

Asked 16 years, 4 months ago Modified 11 years, 3 months ago
Viewed 6k times

22

Another discussion (we've been having a lot of them these days!) in our work is whether data binding is a bad idea or not.

Personally, I think it is a Bad Thing™.

My reasons are thrice:

1. It circumvents my well architectured MVP framework - with databinding, the view communicates bi-directionally with a model. Ewww.
2. It promotes hooking up view controls to datafields at design time. In my experience, this leads to vital code (binding column A to Field X) being obscure and hidden away in some designer file. IMO this code should be explicit and in-your-face, so that it is easy to modify and see what is going on, without having to use a clunky designer interface.
3. Relating to Point #1 this direct binding makes it harder to isolate each component (view, model, controller/presenter) and unit-test.

The pros are that it is easy to set up, and you can take advantage of some nice features (validation etc) which come with the plumbing already done for you.

But for me, databinding becomes much more of a hindrance when dealing with a large data-centric application.

Any thoughts?

data-binding

Share

Improve this question

Follow

edited Jul 11, 2012 at 19:15

BoltClock

722k ● 165 ● 1.4k ● 1.4k

asked Aug 21, 2008 at 8:25

Duncan

319 ● 3 ● 6

10 Answers

Sorted by: Highest score (default)

6

As we say in the UK, *"It's Horses for courses"*

First off all, I agree with you! But...

For enterprise level applications, then spending the extra time on the system architecture, modelling and standards will give you a robust and sustainable system.

But it will take longer to develop (or at least longer to get to an initial release) and this may not be appropriate for every system or every part of the system.

Sometimes you just need to "get it done and done quick". For internal applications, back office systems and maintenance applications that are rarely used or very dynamic (the spec's change often) then there is little justification in building the Rolls Royce solution for this. It's better to get the developer spending time on the CRITICAL part of the system.

What you have to avoid / prevent is using these "one click framework" solutions on the MISSION CRITICAL area's of the system where the big transaction rate area's and where data quality and integrity is critical. Spend quality time shaving the milliseconds off on the most heavily used area's on the system!!

Share Improve this answer

Follow

answered Aug 21, 2008 at 8:38

Guy

9,826 ● 7 ● 39 ● 43

4

Another discussion (we've been having a lot of them these days!) in our work is whether data binding is a bad idea or not.

Personally, I think it is a Bad Thing™.

Strong opinion, but imho, you bring out all the wrong reasons.

1. > It circumvents my well architectured MVP framework - with databinding, the view communicates bi-directionally with a model. Ewww.

   I guess it depends on the implementation of the data binding. In the early years of my programming career, I used to do a lots of VBA for MS Access programming and Access forms had indeed this direct binding to tables/fields in database.

   Most of the general purpose languages/frameworks have databinding as a separate component, do not use such a direct binding and are usually considered as a easy generic dropin for a controller in MVC pattern sense.

2. > It promotes hooking up view controls to datafields at design time. In my experience, this leads to vital code (binding column A to Field X) being obscure and hidden away in some designer file. IMO this code should be explicit and in-your-face, so that it is easy to modify and see what is going on, without having to use a clunky designer interface.

   I guess you are talking about the binding in WinForms?

My experience with win forms comes from a long ago, so I might be pretty out of date here. It sure is a convenience feature, and I would strongly argue against it, unless you are writing really simple modal context CRUD style interfaces.

3. > Relating to Point #1 this direct binding makes it harder to isolate each component (view, model, controller/presenter) and unit-test.

   Again - assuming the view (a widget in WinFoms?) is tied together with databinding awareness, you are right.

> But for me, databinding becomes much more of a hindrance when dealing with a large data-centric application.

Quite contrary - if data binding is implemented as an independent component (eg. bindings in Cocoa or JFace DataBinding, or JGoodies Binding), that acts as a controller between View and a Model, taking care of all the nitty-gritty of event handling and conversion and validation, then it is just so much more easier to use, change and replace than your custom controller code doing just the same thing.

The only downside of a general purpose data binding framework is that if the binding is off and/or misconfigured, the interactions between bound pieces are

just notoriously difficult to debug due to the level of abstraction inside the data binding code... So You better not make any mistakes! ;)

Share Improve this answer

Follow

edited Jul 11, 2012 at 19:18

BoltClock

722k ● 165 ● 1.4k ● 1.4k

answered Aug 21, 2008 at 10:41

Roland Tepp

8,511 ● 11 ● 56 ● 73

3

I've used databinding in some pretty large systems and find that it works pretty well.

Seems that I do things a bit differently from you though ...

... I don't databind to the model, instead to a dedicated view class that works as an adapter between the model's structure and what I need on screen. This includes things like providing choices for comboboxes & listviews, and so on.

... I never set up the binding using the UI. Instead, I have a single method (usually called `Bind()` or `BindXYZ()` that hooks everything up in one place.

My Model remains agnostic, knowing nothing about databinding; my Presenter sticks to the workflow coordinate it's designed for; my Views are now also simple classes (easy to test) that encapsulate my UI

behavior (is button X enabled, etc) and the actual UI is relegated to a simple helper on the side.

Share Improve this answer

Follow

answered Mar 19, 2009 at 19:21

Bevan

44.3k ● 10 ● 84 ● 133

3

I have had a few unshakable realizations about data binding over the last few years:

1. The claim that the data binding allows for the business and presentation to be designed in isolation of each other is actually really quite far from what actually goes on in reality. Usually the deficiencies in the technologies become readily apparent and then all you have done is break apart the UI from the UI-specific business and the resulting separation often becomes far more unwieldy than a all-in-one approach.
2. Most data binding engines (HTML / WPF / or whatever) all make assertions on the technical business model, and since the designer is not usually equipped to make said assertions, the developer ends up having to touch the view. Not only that, the view shouldn't be making assertions about the business model---if anything, it should be the other way around.
3. Most of the time, the view model / controller / model / view are all "coupled" and then all you have really done is "move code around" rather than just simply

using code behind. With that said, I do find the most pragmatic approach is often to just use data binding sparingly with code behind and forget about MVVM/MVC esque patterns.

4. Developers often put view level concerns on the view model and then start to use data binding as a crutch rather than a proper approach. for example, I have seen so many view models controlling visibility of UI elements.
5. Admittedly, data binding is useful for "small systems". I have observed that the performance, complexity and maintainability dramatically suffer as an application grows in richness.
6. Memory usage techniques with data binding can often become a real hazard. WPF for example uses a LOT of trickery to avoid issues and often developers can still shoot themselves in the foot. Unless you are using something like Sencha for HTML (I think), you will find your memory foot print on your applications start to suffer even with a modest amount of data.
7. I have found that data binding / UI patterns in general sometimes tend to break down a little when dealing with hierarchical and situational data / presentation.

My personal outlook on data binding is that it is a tool that can be easily abused yet has some compelling uses. You can say the same for any technique, pattern, or guideline. Like anything, too much of something tends to become a

problem. I tend to like to try and use the most pragmatic approach for the situation. Prefer consistency when it is pragmatic to do so, but consistently be pragmatic. In other words, you don't have to go down the path of developing for two years and only then come to the conclusion that the code base has become a grotesque smelly mammoth in a china shop full of orphan kittens.

...

Share Improve this answer

Follow

answered Sep 10, 2013 at 5:46

Garth Pickell

71 ● 1

2

@Point 1: Isn't the data binding engine the controller, if you really want to think in patterns? You just do not program it yourself, which is the whole point of using data binding in the first place.

Share Improve this answer

Follow

answered Aug 21, 2008 at 8:34

Timbo

28k ● 11 ● 51 ● 75

1

No. DataBinding *when used correctly* is a Good Thing™.

1. No; but see #2 and #3. Make the Presenter expose the properties/well-defined sources to bind. Do not expose the Model. Nothing is circumvented.

2. I agree. I do not use *any* of the standard ASP.NET data-sources. Instead I use GenericDataSourceControl which is wired to a "select method" that returns well-defined types. The DataSource consumers in the View only knows of these Presenter-types; nothing more.
3. No. Relating to #1. The Presenter exposes the properties/well-defined sources to bind. These can be tested without the view for correctness (unit tests), and with the view for correctness of application (integration tests).

(My Experience is using ASP.NET WebForms, which may differ from other data-binding scenarios.)

Share Improve this answer Follow

answered Jul 11, 2012 at 19:10

user166390

0

@Timbo:

Yes and no.... but from a TDD perspective I'd like to cordon-off each controller so that I can test it in isolation. Also, say we want to run each edit via an EditCommand (so that we support Undo, for example) - for me, this rules out databinding.

@Guy:

Yes, this is exactly my POV. For me, databinding is great for very simple apps, but we don't do any of those!

Share Improve this answer

Follow

answered Aug 21, 2008 at 8:46

Duncan

319 • 3 • 6

Data binding does not necessarily rule out command based editing support. There is an implementation of JFace DataBinding for EMF that dispatches all the edits through command framework. I have not used that, but I've heard reports that it works perfectly. – Roland Tepp Sep 25, 2008 at 13:44

0

I feel that in many frameworks, data binding is just an excuse to do things the easy way. It often results, as does almost any designer-generated code, in too much code which is too complicated and can't be easily tweaked. I've never come across a task I couldn't do just as well (if not better) and, in most cases, just as quickly, by data binding as by writing the code myself.

Share Improve this answer

Follow

answered Sep 18, 2008 at 7:05

J D OConal

624 • 4 • 14

0

I have used databinding on large enterprise systems inconjunction with a framework. In my case it was CSLA.

It worked so well, and was extremly fast to get the view working. CSLA has lots of support for databinding and validation built in though.

If it breaks the MVP patturn, so what? if it works better and faster and is easier to manage. However, I would argue that it doesn't break the patturn at all... You can hook up databind in the presenter as it has a reference to the view and also to the model.

for example this is what you would put in your presenter and it would populate the list box or whatever control you want.

myView.list.datasource = myModel.myCollection;

- Also I would like to point out the databinding shouldn't be taken as an all or nothing approch. Many times I use databinding when i have a simple and easy UI requirment to map to my object model. However, when there is special functionality needed I might put some code in the presenter to build up the view as I need it rather than using databinding.

Alan

answered Feb 18, 2009 at 13:12

Alan Johnson

0

I quite agree with you, data binding have drawbacks... In our application, if not used carefully, it leads us sometimes to bad data consistency...

But there may be some elegant ways work with databinding with large forms?

Please give me your opinion here: How to use a binding framework efficiently

Share Improve this answer

Follow

edited May 23, 2017 at 11:53

Community Bot

1 ● 1

answered Jul 29, 2011 at 8:57

Sebastien Lorber

92k ● 73 ● 297 ● 427