How to convince my co-workers not to use datasets for enterprise development (.NET 2.0+)

Asked 16 years, 3 months ago Modified 12 years, 4 months ago

Viewed 2k times

13

Everyone I work with is obsessed with the data-centric approach to enterprise development and hates the idea of using custom collections/objects. What is the best way to convince them otherwise?

oop

Share

Improve this question

Follow

edited Jul 27, 2012 at 7:59

ThinkingStiff

65.3k ● 30 ● 147 ● 241

asked Sep 1, 2008 at 2:16

Toran Billups

27.4k ● 41 ● 158 ● 272

16 Answers

Sorted by: Highest score (default)

Do it by example and tread lightly. Anything stronger will just alienate you from the rest of the team.

21

Remember to consider the possibility that they're onto something you've missed. Being part of a team means taking turns learning & teaching.

No single person has all the answers.

Share Improve this answer

Follow

answered Sep 1, 2008 at 2:39

Mark Biek

151k ● 54 ● 158 ● 201

10

If you are working on legacy code (e.g., apps ported from .NET 1.x to 2.0 or 3.5) then it would be a bad idea to depart from datasets. Why change something that already works?

If you are, however, creating a new apps, there a few things that you can cite:

- Appeal to experiencing *pain* in maintaining apps that stick with DataSets
- Cite performance benefits for your new approach
- Bait them with a good middle-ground. Move to .NET 3.5, and promote LINQ to SQL, for instance: while still sticking to data-driven architecture, is a huge, huge departure to string-indexed data sets, and enforces... voila! Custom collections -- in a manner that is hidden from them.

What is important is that whatever approach you use you remain consistent, and you are completely honest with

the pros and cons of your approaches.

If all else fails (e.g., you have a development team that utterly refuses to budge from old practices and is skeptical of learning new things), this is a **very, very clear sign** that you've outgrown your team it's time to leave your company!

Share Improve this answer

Follow

answered Sep 1, 2008 at 3:09

Jon Limjap

95.3k ● 15 ● 103 ● 153

1 Promote Linq2SQL? L2S is dead - promote EF instead. – stephbu Nov 8, 2008 at 15:22

2 @stephbu - and EF is under heavy revision, so neither (in current form) is a good bet long term. But L2S requires less investment / complexity, so (given the current flux) that makes it more attractive than EF IMO. – Marc Gravell Nov 9, 2008 at 20:53

3 let's see...L2S is dead...EF is under heavy revision... hey! datasets seem to be pretty stable, maybe we should use them! ;-) – Steven A. Lowe Nov 19, 2008 at 2:06

2 "Linq to SQL is dead" seems to be an web meme these days. Like most web memes, no one knows where it came from, and it is nonsensical. – Ryan Lundy Jan 28, 2010 at 19:39

▲

9

> *Remember to consider the possibility that they're onto something you've missed. Being part of a team means taking turns learning & teaching.*



Seconded. The whole idea that "enterprise development" is somehow distinct from (and usually the implication is 'more important than') normal development really irks me.

If there really is a benefit for using some technology, then you'll need to come up with a considered list of all the pros and cons that would occur if you switched.
Present this list to your co workers along with explanations and examples for each one.

You have to be realistic when creating this list. You can't just say "Saves us lots of time!!! WIN!!" without addressing the fact that sometimes it is going to take MORE time, will require X months to come up to speed on the new tech, etc. You have to show concrete examples where it will save time, and exactly how.

Likewise you can't just skirt over the cons as if they don't matter, your co-workers *will* call you on it.
If you don't do these things, or come across as just pushing what you personally like, nobody is going to take you seriously, and you'll just get a reputation for being the guy who's full of enthusiasm and energy but has no idea about anything.

BTW. Look out for this particular con. It will trump everything, unless you have a *lot* of strong cases for all your other stuff:

- Requires 12+ months work porting our existing code. You lose.

Share Improve this answer

Follow

answered Sep 1, 2008 at 3:00

Orion Edwards
123k ● 66 ● 245 ● 339

7

Of course, "it depends" on the situation. Sometimes DataSets or DataTables are more suited, like if it really is pretty light business logic, flat hierarchy of entities/records, or featuring some versioning capabilities.

Custom object collections shine when you want to implement a *deep hierarchy/graph* of objects that cannot be efficiently represented in flat 2D tables. What you can demonstrate is a large graph of objects and getting certain events to propagate down the correct branches without invoking inappropriate objects in other branches. That way it is not necessary to loop or Select through each and every DataTable just to get the child records.

For example, in a project I got involved in two and half years ago, there was a UI module that is supposed to display questions and answer controls in a single WinForms DataGrid (to be more specific, it was Infragistics' UltraGrid). Some more tricky requirements

- The answer control for a question can be *anything* - text box, check box options, radio button options, drop-down lists, or even to pop up a custom dialog box that may pull more data from a web service.
- Depending on what the user answered, it can trigger more sub-questions to appear directly under the parent question. If a different answer is given later, it

should expose another set of sub-questions (if any) related to that answer.

The original implementation was written entirely in DataSets, DataTables, and arrays. The amount of looping through the hundreds of rows for multiple tables was purely mind-bending. It did not help the programmer came from a C++ background attempting to *ref* everything (hello, objects living in the heap use *reference variables*, like pointers!). Nobody, not even the originally programmer, could explain why the code is doing what it does. I came into the scene more than six months after this, and it was stil flooded with bugs. No wonder the 2nd-generation developer I took over from decided to quit.

Two months of tying to fix the chaotic mess, I took it upon myself to redesign the entire module into an object-oriented graph to solve this problem. yeap, complete with abstract classes (to render different answer control on a grid cell depending on question type), delegates and eventing. The end result was a 2D dataGrid binded to a deep hierarchy of questions, naturally sorted according to the parent-child arrangement. When a parent question's answer changed, it would raise an event to the children questions and they would automatically show/hide their rows in the grid according to the parent's answer. Only question objects down that path were affected. The UI responsiveness of this solution compared to the old method was by orders of magnitude.

 edited Sep 16, 2008 at 14:03

Follow

answered Sep 1, 2008 at 3:12

icelava
9,857 ● 7 ● 53 ● 75

5

Ironically, I wanted to post a question that was the exact opposite of this. Most of the programmers I've worked with have gone with the custom data objects/collections approach. It breaks my heart to watch someone with their SQL Server table definition open on one monitor, slowly typing up a matching row-wrapper class in Visual Studio in another monitor (complete with private properties and getters-setters for each column). It's especially painful if they're also prone to creating 60-column tables. I know there are ORM systems that can build these classes automagically, but I've seen the manual approach used much more frequently.

Engineering choices always involve trade-offs between the pros and cons of the available options. The DataSet-centric approach has its advantages (db-table-like in-memory representation of actual db data, classes written by people who know what they're doing, familiar to large pool of developers etc.), as do custom data objects (compile-type checking, users don't need to learn SQL etc.). If everyone else at your company is going the DataSet route, it's at least technically possible that DataSets are the best choice for what they're doing.

Share Improve this answer

Follow

answered Nov 8, 2008 at 13:58

MusiGenesis

75.3k ● 41 ● 197 ● 338

I can see pros and cons for both options like you mentioned, but I wanted to get the other developers to step away from the DataSet route because it's so tightly coupled with the .NET platform. If we switch to java for example, they might have to spend more time learning about pure OOP concepts
– Toran Billups Nov 11, 2008 at 17:18

I hear you - it's always hard to get people to think about new ways of doing things. I'm sufficiently mired in my ways that my first thought after reading your comment was "I wonder if Java has things like DataSets and DataTables".
– MusiGenesis Nov 11, 2008 at 17:58

I once tricked a co-worker into going the other direction (data objects to DataTables), but the story would surely get me down-voted here, and I don't think the trick is reversible. :)
– MusiGenesis Nov 11, 2008 at 20:11

3

Datasets/tables aren't so bad are they?

Best advise I can give is to use it as much as you can in your own code, and hopefully through peer reviews and bugfixes, the other developers will see how code becomes more readable. (make sure to push the point when these occurrences happen).

Ultimately if the code works, then the rest is semantics is my view.

Share Improve this answer

answered Sep 1, 2008 at 2:25

Follow

Mark Glorie
3,473 ● 3 ● 29 ● 31

2

I guess you can trying selling the idea of O/R mapping and mapper tools. The benefit of treating rows as objects is pretty powerful.

Share Improve this answer

Follow

answered Sep 1, 2008 at 2:40

Eugene Yokota
95.5k ● 45 ● 217 ● 320

> mind you, so is the benefit of treating rows as rows of data instead of forcing them into artificial OO terms of what they actually are. – gbjbaanb Dec 14, 2008 at 15:05

2

I think you should focus on the performance. If you can create an application that shows the performance difference when using DataSets vs Custom Entities. Also, try to show them Domain Driven Design principles and how it fits with entity frameworks.

Share Improve this answer

Follow

answered Sep 1, 2008 at 3:26

azamsharp
20.1k ● 38 ● 146 ● 228

Don't make it a religion or faith discussion. Those are hard to win (and is not what you want anyway)

2

Don't frame it the way you just did in your question. The issue is not getting anyone to agree that this way or that way is the general way they should work. You should talk about how each one needs to think in order to make the right choice at any given time. give an example for when to use dataSet, and when not to.

I had developers using dataTables to store data they fetched from the database and then have business logic code using that dataTable... And I showed them how I reduced the time to load a page from taking 7 seconds of 100% CPU (on the web server) to not being able to see the CPU line move at all.. by changing the memory object from dataTable to Hash table.

So take an example or case that you thing is better implemented differently, and win that battle. Don't fight the a high level war...

Share Improve this answer

Follow

answered Sep 1, 2008 at 7:41

csmba

4,083 ● 3 ● 33 ● 42

2

If Interoperability is/will be a concern down the line, DataSet is definitely not the right direction to go in. You CAN expose DataSets/DataTables over a service but whether you SHOULD or is debatable. If you are talking .NET->.NET you're probably Ok, otherwise you are going to have a very unhappy client developer from the other side of the fence consuming your service

Share Improve this answer

Follow

answered Apr 2, 2009 at 6:27

Abhijeet Patel

6,848 ● 9 ● 54 ● 93

1

You can't convince them otherwise. Pick a smaller challenge or move to a different organization. If your manager respects you see if you can do a project in the domain-driven style as a sort of technology trial.

Share Improve this answer

Follow

answered Sep 1, 2008 at 3:17

liammclennan

5,368 ● 3 ● 35 ● 31

1

If you can profile, just Do it and profile. Datasets are heavier then a simple `Collection<T>`

DataReaders are faster then using Adapters...

Changing behavior in an objects is much easier than massaging a dataset

Anyway: Just Do It, ask for forgiveness not permission.

Share Improve this answer

Follow

answered Sep 1, 2008 at 5:15

Brian Leahy

35.4k ● 12 ● 46 ● 60

Most programmers don't like to stray out of their comfort zones (note that the intersection of the 'most

1

programmers' set and the 'Stack Overflow' set is the probably the empty set). "If it worked before (or even just worked) then keep on doing it". The project I'm currently on required a lot of argument to get the older programmers to use XML/schemas/data sets instead of just CSV files (the previous version of the software used CSV's). It's not perfect, the schemas aren't robust enough at validating the data. But it's a step in the right direction. The code I develop uses OO abstractions on the data sets rather than passing data set objects around. Generally, it's best to teach by example, one small step at a time.

Share Improve this answer

Follow

edited Jul 3, 2012 at 14:17

user142162

answered Sep 1, 2008 at 8:50

Skizz

71k ● 10 ● 74 ● 109

0

There is already some very good advice here but you'll still have a job to convince your colleagues if all you have to back you up is a few supportive comments on stackoverflow. And, if they are as sceptical as they sound, you are going to need more ammo. First, get a copy of Martin Fowler's "Patterns of Enterprise Architecture" which contains a detailed analysis of a variety of data access techniques. Read it. Then force them all to read it.

Job done.

Share Improve this answer

Follow

answered Dec 14, 2008 at 14:51

kotoku

0

data-centric means less code-complexity.

custom objects means potentially hundreds of additional objects to organize, maintain, and generally live with. It's also going to be a bit faster.

I think it's really a code-complexity vs performance question, which can be answered by the needs of your app.

Share Improve this answer

Follow

answered Jul 2, 2009 at 19:53

alchemical

13.9k ● 24 ● 84 ● 111

0

Start small. Is there a utility app you can use to illustrate your point?

For instance, at a place where I worked, the main application had a complicated build process, involving changing config files, installing a service, etc.

So I wrote an app to automate the build process. It had a rudimentary WinForms UI. But since we were moving towards WPF, I changed it to a WPF UI, while keeping the WinForms UI as well, thanks to Model-View-Presenter. For those who weren't familiar with Model-

View-Presenter, it was an easily-comprehensible example they could refer to.

Similarly, find something small where you can show them what a non-DataSet app would look like without having to make a major development investment.

Share Improve this answer

Follow

answered Dec 14, 2009 at 20:24

Ryan Lundy

210k ● 41 ● 184 ● 216

All of that said, Toran, I don't necessarily agree with your premise. Now if they're using untyped DataSets, with a ton of ugly string literals everywhere, then yes, anything would be better. But typed DataSets are a good solid data access technology. Whether one prefers them or an O/R mapper is a matter of style. Don't try to force them into your style; it won't work. Create an example and let them see whether there are clear benefits. – Ryan Lundy Dec 14, 2009 at 20:27