Build your own CAB Part #7 – What's the Model?

Posted by Jeremy Miller on June 6, 2007

First, go catch up on what's come before:

1. Preamble
2. The Humble Dialog Box
3. Supervising Controller
4. Passive View
5. Presentation Model
6. View to Presenter Communication
7. Answering some questions

What's the Model?

I've spent most of the series talking about the View or the Presenter, but the Model piece of the triumvirate has a role to play as well.

A couple years ago I participated in a session on design patterns for fat clients that Martin Fowler was running to collect data for his forthcoming sequel to the PEAA book. One of the topics that came up in conversations afterward was whether or not it was desirable to put the real Domain Model on the client or use a mirror version of the Domain Model that you allow to have some user interface specific functionality. In other words, is the Domain Model pattern applied to the user interface a completely different animal with different rules than the traditional POCO Domain Model in the server? We didn't come up with any kind of a consensus then, and I've never made up my mind since.

As I see it, you have four choices for your Model:

1. **Domain Model Class** – Just consume the real application classes. Many times it's the simplest path to take, and leaves you with the fewest moving parts. My current and previous projects both used this approach with a fair amount of success. In my current project our Domain Model classes implement quite a few business rules that come into play during screen interactions. The downsides to consuming the real Domain Model in the View are that you're binding the View more tightly to the rest of the application than may be desirable and the possibility of polluting the Domain Model with cruft to support INotifyPropertyChanged interfaces and other User Interface needs. Part of the reason I'm perfectly happy to consume Domain Model objects directly on my project is that our screen design doesn't require any special UI support for our custom data binding solution.
2. **Presentation Model** – Even if you're largely following a Model View Presenter architecture, the Presentation Model is still useful. Think of this realistic screen scenario: the real domain objects are an aggregate structure and your View definitely needs some INotifyPropertyChanged-type goo. In this particular case a Presentation Model that wraps and hides the real domain objects from the View is desirable. The Presentation Model probably provides a flattened view of the domain aggregate to make data binding smoother while implementing much of the UI infrastructure code, allowing the domain classes to take the shape that is most appropriate for the business logic and behavior and keeping them from being polluted with UI code. I should not that using a Presentation Model **will potentially add extra work to synchronize the data** with the underlying domain objects.
3. **Data Transfer Object** – Forget about behavior and just use a lump of data. This is often a result of hooking the user interface directly to the return values of a web service. In my previous application we wrote an absurd amount of mapping code to map data transfer objects to domain objects and vice versa and I think we all felt like it ended up being a huge waste (we *really* needed to isolate our client from the backend, so it was mostly justified). This time around I'm honestly quite content just to bind some of the user interface directly to the Data Transfer Objects returned from our Java web services. I feel a little bit dirty about this, but this approach is dirt simple. It does couple us a bit more to the server than I would normally like, but I'm hoping to compensate with a fairly elaborate Continuous Integration scheme that does a fully integrated build with end to end StoryTeller/Fit tests anytime either codebase changes to detect breaking changes.
4. **DataSet/DataTable/DataView** - See below:

Upcoming posts…

How I'm using StoryTeller to tes
FubuMVC
Building a "Lookup" html conver
FubuMVC
FubuMVC's Configuration Mode
Sauce"
Managing Script dependencies
FubuMVC
Authorization and FubuMVC
Continuations
Composing Views with FubuMV
Extensible Model Binding with F
Introducing "Bottles"
Modular Packaging with FubuM
Self-Installing Apps w/ FubuMV
Routing and Behavioral Conven
FubuMVC
What Should I Learn?

Blogroll

Follow me on Twitter
FubuMVC on GitHub
My GitHub Page
StoryTeller on GitHub
StructureMap on GitHub

Archives

Somebody has to ask, what about DataSet's? I despise DataSet's in general, but there's no denying that sometimes the easiest way to solve a problem is to revert back to Stone Age techniques and use them. There's a memorable scene from the first Lord of the Rings movie when Gandalf is speaking to Frodo very dramatically of the One Ring - "It wants to be found." It's the same way in WinForms. The user interface widgets often **want to consume a DataSet**. The entire WinForms UI toolkit was originally wrapped around a very datacentric view of the world and sometimes you just go along with it.

Alright, it's not that bad. I will happily use a DataSet/DataTable/DataView in my user interface as the ostensible Model any time it's the easiest way to use a UI widget or when I want to take advantage of the sorting and filtering capabilities of a DataTable (I think that equation changes when Linq to Objects hits). The Data* classes aren't my real domain though, I generally convert my real classes to a DataTable just in time for display. And no, a DataSet-centric approach doesn't buy me much because as I'll show in the next section the Model has real responsibilities beyond just being a dumb bag of data. Plus the little issue that DataSet's are not interoperable and we're using web services written with Java for our backend services.

While a DataSet makes the data binding generally very simple, you're left with the usual drawbacks to a DataSet. You can't embed any real logic into the DataSet, so you have to be careful with duplication of logic. If you insist on a DataSet approach, I'd recommend giving the Table Module approach some thought as a way to centralize the related business logic for a particular set of data to avoid duplication. Personally, I think DataSet's are clumsy to use inside of automated tests in terms of test setup. A strongly-typed DataSet helps to at least get some Intellisense, but they annoy me as well. Plus you'll often find yourself building data that's meaningless to the test just to satisfy the referential integrity rules of a DataSet. That's wasted effort.

About Jeremy Miller

Jeremy is the Chief Software Architect at Dovetail Software, the coolest ISV in Austin. Jeremy began his IT career writing "Shadow IT" applications to automate his engineering documentation, then wandered into software development because it looked like more fun. Jeremy is the author of the open source StructureMap tool for Dependency Injection with .Net, StoryTeller for supercharged acceptance testing in .Net, and one of the principal developers behind FubuMVC. Jeremy's thoughts on all things software can be found at The Shade Tree Developer at http://codebetter.com/jeremymiller.

View all posts by Jeremy Miller →

This entry was posted in Build your own CAB, Design Patterns. Bookmark the permalink. Follow any comments here with the RSS feed for this post.

← Build your own Cab – Answering some questions

Ian Cooper reviews Linq to SQL in regards to DDD and TDD →

6 Comments The Shade Tree Developer Login

Sort by Best Share Favorite

Join the discussion…

Jeremy D. Miller • 7 years ago

Ralf,

I've most certainly read their paper. It's all the same stuff I've been talking about though. They're taking a Passive View approach with specific adapters between the Presenter and View for screen synchronization (which I would whole heartedly recommend if you go down the Passive View approach). The communication is through events instead of direct calls to the Presenter the way I prefer.

• Reply • Share ›

Ralf Kretzschmar • 7 years ago

Hi Jeremy,
did you now about this: http://atomicobject.com/pages/...
There are also two interviews with the guys behind Presenter First on channel nine / AR Cast.
Don't get me wrong, I really appreciate the things you have brought to us in the last couple of weeks, but I really think Presenter First is the best idea I came accros in years.

• Reply • Share ›

joeyDotNet • 7 years ago

One of the techniques I've used lately and have been successful with is creating screen-specific DTOs. This is a tip I picked up from JP when he did his Nothin' But .NET class down here in VA.

Now I've only done this on a couple web projects (both using MonoRail of course... :), but I've found it great in these scenarios, especially if you can leverage a UI framework that will automatically 2-way databind your DTOs and views, like MonoRail.

• Reply • Share ›

Paul Rayner • 7 years ago

Another issue related to the use of a Dataset/Datatable that recently has come up for me is the handling of null date values. If you want to leave a date value blank in your form then when you copy the null value into the DataTable it will (by default) throw an exception. I found a way around this, but it was extra head-scratching that I would rather have not had to do.

In other words, while this can sometimes be a worthwhile approach, there may extra baggage with using a DataTable that must also be considered.

• Reply • Share ›

Alex Scordellis • 7 years ago

In the past I've created strongly-typed DataSets to represent my domain model and used .NET 2.0's partial classes to add functionality to them. This allows me to take advantage of the easy binding, navigation etc. of DataSets and still have an OO domain model where the behaviour lives in the same class as the data. Of course this stops you from using inheritance and other OO features in your model, and you still have the problems related to satisfying referential integrity etc, but for the small project I was working on it was a neat solution.

• Reply • Share ›

March 2007
February 2007
January 2007
December 2006
November 2006
October 2006
September 2006
August 2006
July 2006
June 2006
May 2006
April 2006
March 2006
February 2006
January 2006
December 2005
November 2005
October 2005
September 2005
August 2005
July 2005
June 2005
May 2005
April 2005

What others have said…

Duncan on What an Amazing (C You've Discovered

paweł paul on FubuMVC Learns Spark

paweł paul on I'm looking for so testers for StoryTeller (OSS tool testing)

paweł paul on Serenity

paweł paul on My Programming

Home
About
CodeBetter CI
Community
Editors
Search Results