

Re: Databinding – Best Practice (object-oriented)

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework.windowsforms.databinding/2004-06>

From: Grant Frisken (gfrisken.minusspam_at_optusnet.com.au)

Date: 06/08/04

Date: Wed, 9 Jun 2004 09:20:39 +1000

"Alfredo Novoa" <alfredo@nospam.es> wrote in message
news:40c57f47.918600@msnews.microsoft.com...

> On Tue, 8 Jun 2004 08:52:37 +1000, "Grant Frisken"

> <gfrisken.minusspam@optusnet.com.au> wrote:

>

>

> No you can not. What I mean is that tables are not "first class
> citizens" and arrays are.

>

> What I mean is something like this:

>

> var a relation { a integer, b integer } key { a };

>

> a := relation { tuple { a 1, b 2 }, tuple { a 2, b 1 } };

>

> var b relation { a integer } key { a };

>

> b := relation { tuple { a 1 }, tuple { a 3 } };

>

> b := (a join b) { a };

>

> With ADO.NET you can not do anything of this.

You can do all of this – but it takes a lot more code.

> > Using a

> > relational model throughout your application severely restricts the

> > developers ability to use some very powerful OO features (such as

> > inheritance, interfaces).

>

> This is not true. The Relational Model is completely independent of
> the inheritance. There are relational languages with inheritance and
> polymorphism, and all of them have encapsulation.

Do any of the major RDMS vendors currently support these features and have
an interface into .NET for this?

>
>>> *ADO.NET is a clear backward step in ease of use, and it has important
>>> problems if you want to map big tables.*
>>
>>*I agree. The last .Net project I worked on I used ADO instead of
ADO.NET
>>for this very reason. The complete abandonment of any useful connected
>>model seems to be a big mistake. If you have a query that may
potentially
>>return a large dataset then ADO.NET is hopeless because it loads the
entire
>>dataset into memory before it can do anything else. This is
catastrophic
>>if you have a client/server design because the entire dataset is shipped
>>over the wire.*
>
>*I can't agree more. For me ADO.NET is simply unusable for complex
> applications.*
>
>*I use it for Pocket PC apps only because it does not worth to create a
> replacement for very simple applications.*
>
>>*Well I agree OO is very illdefined and it is easy to get into a war of
"I'm
>>more OO than you". But I think that there would be general agreement
that
>>relational model lacks many of the features (such as inheritance,
interfaces
>>etc) that make OO such a powerful tool for application development.*
>
>*But IMO that agreement is based on the ignorance of what The
> Relational Model really is. The Relational Model is not SQL. SQL was a
> quick and dirty experiment of the 70's that never should be released.*
>
Perhaps – but it is the now widely adopted standard. You will find it
difficult to persuade developers to ditch the database independence this
gives them (although far from complete or perfect) and bind themselves
tightly to one vendors view of the world. I think this is part of the
problem that OO databases have had in getting developers to adopt them.

> *You can have single and multiple inheritance with The Relational
> Model, and even you have inheritance in SQL 3.*
>
>>> *The correct equation is:
>>>
>>> one class = one domain.
>>> Anything else is a great blunder.*
>>
>>*Not sure what you mean by a domain.*
>
>*A data type. What SQL domains should be but they are not.*

>
> *I suppose you know what SQL domains are:*
>
> *CREATE DOMAIN name [AS] data_type*
> *[DEFAULT expression]*
> *[constraint [...]]*
>

I have to plead ignorance here. I am by no means a DBMS guru – but I was unable to find any reference to this in MSDN – so I am assuming that SQL Server does not support them. From the documentation I could find on sql.org it looks like this is restricted to defining primitive datatypes with constraints and defaults.

To claim that this is the only possible mapping to classes seems to me to be like designing a language which only supported classes derived from primitive types.

> *Class and domains are the same.*
>
> > *I think you may be being a little dogmatic.*
>
> *No, it is basic common sense. The most common meaning of class is data*
> *type (class may mean many things like most OO terms), and database*
> *domains are data types. They are the same.*

But OO classes do not equal primitive types. In fact in general an OO class equals type consisting of member variables and member functions

> *On the other hand tables are variables and it is evident that to*
> *confuse a variable with a type is a monstrous blunder.*

But table definitions or schemas are not variables and could easily be equated with classes (although one to one mappings are probably a mistake as we have discussed)

> *User interface logic is application development, but business logic is*
> *data management.*
>
> *Business logic are the data integrity and data derivation rules.*
> *Business logic and databases are indissoluble. Business logic is ALL*
> *in database design.*

Well this depends on your definition of business logic. I'd be more than happy to use the database to enforce as much business logic and constraints as possible. However in my experience there are some things that I would classify as business logic that you can't do in a RDMS.

Regards
Grant