

Re: Remobjects v KBM

Source:

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.thirdpartytools.general/2004-02/1324.html>

From: Lauchlan M (LMackinnon_at_NOSPAMHotmail.com)

Date: 02/20/04

Date: Fri, 20 Feb 2004 09:10:16 +1000

> > *In DA, you start with*
> > *defining the data connections and data definitions 'datasets' you are*
going
> > *to use. (Although I have not yet got my head around how to start with*
the
> > *abstract data definitions and map them to the actual data sources. At*
the
> > *moment I do it by dragging from actual sources in the schema modeller,*
so if
> > *anyone can tell me how to start with defining the data at an abstract*
level
> > *and _later_ map it to actual data sources I'd appreciate that!) and then*
the
> > *business processors (like kbmmw resolvers) and data tables (like kbmmw*
> > *client query components) follow from that.*
>
> *Ok.. I can see this as an interesting new designtime tool for kbmMW...*
drag and drop of queries etc. from a datasource
> *repository, automatically creating the components needed. Actually the*
wizard have a good part of that already, but it
> *doesnt let you define the SQL at that time. This is done at a later stage.*

In DA, you can define a dataset abstractly, for example a dataset 'customers' could be abstractly defined, without reference to any actual tables or fields in actual database servers, say by defining fields customerID, CustomerFirstName, CustomerLastName, and CustomerPhoneNumber. Then, connections can be created to say SQL Server, Oracle, Interbase and NexusDB databases, specifically to tables within those databases which have fields analogous to the above (not necessarily with the same name). Mappings can then be created from the abstract dataset definition in 'customers' to actual tables in each of the above databases. This is all visually very RAD in the schema modeller.

> > *Both the server side and client*
> > *side things follow from the abstract definition you start with, whereas*
in

- > > *kbmmw you choose appropriate server side and client side components and*
- > > *hook/code them up appropriately. So architecturally, it does not seem to*
- me
- > > *to be as driven by the central abstract definitions (of data, services).*
- >
- > *Well.. in kbmMW I suggest people to not use client side SQL of abstraction*
- reasons, but instead use what has been
- > *defined on the server using the named query setup.*

And security reasons! <g> But I don't think doing the above in DA involves client-side SQL. What you do is retrieve a dataset from the server using a service you code for that. In the server-side implementation of the service you don't necessarily explicitly use SQL directly either (but code it implicitly – this makes your code not be database connection specific). The way this works and why it is good are detailed in the public DA web articles DA01 and DA03 on the remobjects website.

- > *The client can even poll the server to get information about what*
- > *named queries are available, and can then request their definitions.*

Yep, similarly in DA the client can request the schema definitions.

- > *Hence its defined serverside, unless people*
- > *absolutely want to do it client side which kbmMW also supports. I can see*
- however that I might want to create a couple
- > *of pull down menus to make named query selection (by listing server side*
- published named queries) easily available
- > *during designtime. Noted.*

That's a good idea!

But I think you might be missing the point of how great a wizard the schema modeller is! <g> I'm sure Alessandro and Marc would be happy for you to continue to miss this point on this (! <g>) . . . but if you have a look through say the publicly available RO/DA DA01, DA02 and DA03 web articles I'm sure you'll get a sense of what this tool offers. I'm sure it must be 'kocher' for you to read each other's published documents, unless you have some gentleman's agreement otherwise. But maybe they log your address from your visits and rib you about it afterwards! <g>

- > > *Another area where from a use-point-of-view the RO/DA architecture*
- shines is
- > > *in its 'wizards', particularly the service builder and the schema*
- modeller,
- > > *which allow you to specify your application on an abstract level very*
- > > *cleanly before diving in and coding.*
- >
- > *As I mentioned I have only looked 'over shoulders' but isnt it basically a*
- TLB type editor with a rather basic code
- > *generator?*

I'm not sure what you mean, but if you have a detailed look through the DA06 article (strongly typed data tables), you will see that the starting point in the article is the schema 'customers' dataset. Based on the abstract interface ICustomers, on the server side then business processor classes (like kbmmw resolvers) are coded up to resolve the client deltas back to the server, and on the client side dataset-specific getters and setters are coded up / generated for the DataTable (functions roughly analogously to the kbmmwclientquery, except strongly typed). Business logic is defined by descending from ICustomers and adding business logic there, which can then be called from the client.

So in this example from the RO/DA website, you can see that it starts with the abstract data definitions, which is done in a very visually RAD way using the schema modeller, but that this abstract definition also drives the coding of the server side resolving logic and of the client side getters and setters for the 'clientquery', and the definition of business logic related to the dataset (in this case customers) which then sits on the server and can be called from the client.

> > *I know kbmmw has wizards that one might say does similar things, but I think*
> > *that there are differences. I think for example that in RO/DA abstraction*
> > *tools allow you to specify the datasets at an abstract level, and this*
> > *abstract definition drives the development of other parts of the app*
> > *architecture in a very OO way, for example one can hang middle-tier business*
> > *logic off these abstract definitions of the data.*
>
> *Im not sure I understand. What exactly is the abstraction level of the dataset?*

It is defining the dataset abstractly, ie without necessarily any reference to any actual tables in actual database servers. Like if you were sitting down to design a database and you said ok, I'll need a customers table, an orders table etc and you spelt out the fields and fieldtypes you'd need in each table, but you hadn't created any database tables yet.

In DA you can take this abstract definition and put it in the schema modeller, then make connections to actual databases and map these abstract definitions to actual table fields. The actual table fields could be called something quite different. But the rest of DA can go ahead and proceed to develop the app based on of these abstract definitions of the dataset(s), assuming at some point an actual connection and mapping to fields in an actual database will be made and used.

> *We unfortunately dont have any partners in Australia, although we have quite many customers from there.*
> *Thus we would ofcourse seriously be interested in getting an Australian partner.*

borland.public.delphi.thirdpartytools.general: Re: Remobjects v KBM

Good luck with that!

Lauchlan M