

Re: What multi-tier components to use

Source:

<http://coding.derkeiler.com/Archive/Delphi/borland.public.delphi.thirdpartytools.general/2005-05/msg00977.html>

- *From:* "Lauchlan M" <LMackinnonAT_NoSpam_ozemailDOTcomDOTau>
 - *Date:* Fri, 20 May 2005 19:19:51 +1000
-

> Yes. Enterprise Java Beans.

ok, thought it might be.

- > Well, I worked with Java and I didn't like that there were really a lot of
- > interfaces that I had to keep in sync on client- and on the server-side
- for
- > Java Clients to be able to talk to a Java Service.

Well, I guess RO is like that too. Doesn't seem to be a real problem though, as long as you add new functions on the server and don't change the definitions for the old ones.

- > Not to mention that
- > writing the java client was a p*i*t*a even without this "problem".

I've not used Java. I hear it's not as productive as Delphi though.

- > I think (I don't want to argue with anyone about this, this is just my
- > opinion) that the most elegant way of solving the remote
- > procedure/function/method call is to allow people to pass any structure to
- a
- > remote procedure/function/method and to return any structure as a result.
- > Then, let the procedure itself decide what is good and what's not from
- those
- > parameters and let the caller decide if it likes the result. You can
- always
- > throw an exception if you don't like the data you receive (which should be
- > passed over to the caller side).

That sounds reasonable.

I guess the only difference between RO and kbmmw is that RO asks you to define those structures first.

- > Anyway .. not having to sync the interface
- > with every change in the parameter list, you could make your new server
- code

Re: What multi-tier components to use

- > compatible with old clients, while you can still add new features, even if
- > it's by additional parameters for the same methods being used from your
- > clients.

But surely if you have an old client and the interfaces are still valid on the new server, you'd be ok? I would have thought you'd only get into trouble when you get rid of some of the old definitions or change their structure (eg add new parameters or remove some).

- > The "problem" (if you could call it a problem) with that approach
- > is that there is nothing forcing the client to use the same parameter types
- > or even the same parameter list as expected from the server. But, as Kim
- > already said, if you really want to do that (enforce the interface), you can
- > still define your class which will map your predefined method calls to the
- > flexible transport model (even use overloaded methods to do that). And when
- > using interfaces, if your client- and server-side interfaces are out of sync
- > (this is most likely to happen when you add a method to your existing
- > remoting class), your client-side code would still compile and "run", but
- > you would not be able to call the remote methods from the client, since you
- > would get exceptions (if you're lucky, those exceptions would tell you
- > what's wrong). And every old client would stop working, which could have
- > been avoided by using flexible structures with careful coding.

Thanks for your explanation. I don't want to get into an argument either (:)), but I guess I don't see why every client would stop working with a new server, except with poor architecting of the changes.

Anyway, I think you (and Kim) are nicely articulating where kbmmw can have its advantages.

I think this is fairly much a 'horses for courses' kind of issue. Having run on both the kbmmw and RO courses, I can know where my preferences lie, but certainly respect that others may have different preferences.

For me, apart from the technical arguments, that we could go 'to and fro' about <g>, perhaps the biggest argument is that I enjoy programming in RO/DA more (and, for the record, I enjoyed programming in kbmmw more than in ASTA). I think you have to enjoy what you're working with if you want to stick around doing it.

And what people enjoy most [I'm talking programming here, keep your mind on track!] is a personal thing, depending on where their coming from and other factors.

Regards,

Lauchlan Mackinnon

-
- **Follow-Ups:**
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Danijel Tkalcec

 - **References:**
 - ◆ **What multi-tier components to use**
 - ◇ From: Petros Amiridis
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Uffe Kousgaard
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Herbert Sitz
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Lauchlan M
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Lauchlan M
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: C4D – Kim Madsen
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Lauchlan M
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: C4D – Kim Madsen
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Danijel Tkalcec
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Lauchlan M
 - ◆ **Re: What multi-tier components to use**
 - ◇ From: Danijel Tkalcec

 - Prev by Date: [A window explorer](#)
 - Next by Date: [Re: What multi-tier components to use](#)
 - Previous by thread: [Re: What multi-tier components to use](#)
 - Next by thread: [Re: What multi-tier components to use](#)
 - Index(es):
 - ◆ [Date](#)
 - ◆ [Thread](#)