

## Re: Finally which ORM tool?

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-10/msg01769.html>

---

- *From:* Jon Skeet [C# MVP] <[skeet@xxxxxxxx](mailto:skeet@xxxxxxxx)>
  - *Date:* Fri, 12 Oct 2007 18:33:37 +0100
- 

James Crosswell <[james@xxxxxxxxxxxxxxxx](mailto:james@xxxxxxxxxxxxxxxx)> wrote:

Jon Skeet [C# MVP] wrote:

In pure ADO.NET you wouldn't have had an object tree which was meant to represent the same state as the server, so the aliasing wouldn't have been apparent.

Wouldn't you? What kind of things would you be passing back and forth between your n-tier application server and client in an pure ADO.NET implementation of the server? Did you pass datasets back and forth? I sure didn't.

So if you *have* got an object tree, do you just not care about aliasing at all? Often it's not a case of returning results to the client – it's manipulation at the server end which may well being sensitive to aliasing issues.

<snip>

I'm not sure if you're playing the devils advocate or if you're just generally hostile to change but it seems to me that what I'm suggesting is certainly an improvement over the existing frameworks which REQUIRE me to write the above workaround because they REQUIRE sessions, even though these are not required in order for these ORMs to do their work.

The decision to be session-based or not is pretty fundamental to an ORM. If you don't want to use sessions, use an ORM which isn't session-based, such as LLBLGen – but please don't try to suggest that no ORMs should use sessions and manage aliasing and identity issues just because they happen not to be issues for you.

That's the great thing about having a choice of frameworks. I have to say that the fact that most ORM frameworks I've come across *do* have

Re: Finally which ORM tool?

identity management suggests that the requirement for it isn't as rare as you seem to be making it out to be. If sessions have no benefit for most people, why do you think the framework developers include them?

I've done plenty of n-tier work, and it's *\*not\** the opposite for n-tier frameworks. I suspect it depends on what you really want the ORM system to do for you, but I've never regarded the session handling as a problem, especially as Hibernate/nHibernate provide the ability to "re-attach" an object from a previous session into a new one when you want to.

I said sessions created a problem in n-tier (where they don't in desktop apps) and if you've done a lot of n-tier work then you must realize this is the case, since sessions in n-tier typically get created and destroyed on a per thread basis and threads get created/destroyed on a per request basis... requiring the workaround I described above in order to effect updates.

You need to distinguish between the idea of a per-request session and a "per user transaction" session. Again, Hibernate in Action deals with this very well.

I wasn't aware of the re-attach behavior in nHibernate though... that sounds interesting and would at least reduce the amount of needless code I write to one line. I'll try to track down some info on this on the net (although one of my projects is using XPO and they don't have such a "feature"). Thanks for the tip.

I can't say I've looked at whether NHibernate itself has this, but I should imagine it does given how it's present in Hibernate.

The Hibernate authors have put in a *\*lot\** of thought when it comes to this kind of thing, and they're certainly *\*not\** just thinking of desktop client/server apps.

—  
Jon Skeet – <skeet@xxxxxxxx>  
<http://www.pobox.com/~skeet> Blog: <http://www.msmvps.com/jon.skeet>  
If replying to the group, please do not mail me too