

## Re: Finally which ORM tool?

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-10/msg01849.html>

---

- *From:* Jon Skeet [C# MVP] <[skeet@xxxxxxxx](mailto:skeet@xxxxxxxx)>
  - *Date:* Sat, 13 Oct 2007 19:26:26 +0100
- 

Frans Bouma [C# MVP] <[perseus.usenetNOSPAM@xxxxxxxx](mailto:perseus.usenetNOSPAM@xxxxxxxx)> wrote:

Let me rephrase then:

It is easier to write an ORM system which has a rich session system if you force sessions to be used everywhere: it means that every layer of the ORM framework code can rely on sessions being available.

Using a session makes an o/r mapper easier to write, as you can just pick Scott Ambler's design and start typing code. The thing is that as soon as you create a distributed system, you run into problems which have to be solved by the user of the framework and not the framework itself. Re-attaching entity objects from a previous session is 'ok' but you won't know which fields have changed.

You won't know which fields have changed unless you query the database whatever you do – sessions don't make the situation any worse here.

Also, if you use a distributed system, and you want on the `_client_` to have unique objects, you're in for a lot of pain, as you don't have that option, you have to write your own uniquing code on the client (as the session lives on the server!).

Indeed.

With in-memory context objects you solve this, and with a session-less design in the o/r mapper you solve the re-attach crap.

I suspect (although I haven't tried it) that using the ORM system on the client would let you effectively have a "fake" session that you can attach the objects to in the same way as using "context objects".

## Re: Finally which ORM tool?

So why people still find it necessary to design an o/r mapper around a session is beyond me. Sure, it's more work to do it session-less, but the point of a framework isn't the joy it gives to the author of it, but the joy it gives to the user of it.

Well, given how many frameworks \*do\* use it, including organisations which certainly have enough time and money to avoid using sessions gratuitously, I suspect there are advantages which either you don't appreciate the importance of or are unaware of.

Certainly when I've used Hibernate I haven't found the session stuff to be a pain, and it's been nice to be able to treat it as a context for that request.

If you don't want to use sessions, I would suggest you use an ORM which makes them optional, understanding that there may well be session-based features which aren't available because the sessions aren't pervasive in that system.

I can't name one feature which isn't possible in these kind of frameworks to be honest.

I can't off the top of my head either – but that doesn't mean such features don't exist. I hope you won't take it as an insult if I suggest that you might be somewhat biased towards session-less ORMs, too :)

I mean a session which is potentially split over multiple client/server requests.

that means the service is stateful. Isn't that a problem waiting to happen, sooner or later?>

never mind services, let's use a simple example of asp.net and postbacks. Exact the same problem.

Either the service is stateful (which can be distributed state, potentially – I believe EJB 3 containers provide this option) or you create a new session, reattach any data you need (whether passed by the client, or whatever) and don't need any server state.

If you need server state, you need server state: having a session based

Re: Finally which ORM tool?

Re: Finally which ORM tool?

or sessionless ORM isn't likely to change that as far as I can see.

(As another couple of data points, LINQ to SQL has the same session bias, and I believe that the ADO.NET Entity Framework does too.)

That's also why they have teh add/detach hell already looming on the horizon (Linq to sql will throw exceptions in these cases when attaching graphs) and together with deferred execution of linq queries, it will give a lot of problems for users who expect A but run into snag B.

You call it "add/detach hell" but I honestly can't say I ever had any problems when working with Hibernate in this way.

Deferred execution is a fundamental (and very useful) part of LINQ which people will just have to understand. It's not mind-bogglingly new, either – using the Criteria API in Hibernate does exactly the same kind of thing, building up a query which can be executed whenever you want it to be.

--

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

.