Sometimes Connected CRUD application DAL

Asked 15 years, 9 months ago Modified 15 years, 9 months ago

Viewed 555 times

4

I am working on a Sometimes Connected CRUD application that will be primarily used by teams(2-4) of Social Workers and Nurses to track patient information in the form of a plan. The application is a revisualization of a ASP.Net app that was created before my time. There are approx 200 tables across 4 databases. The Web App version relied heavily on SP's but since this version is a winform app that will be pointing to a local db I see no reason to continue with SP's. Also of note, I had planned to use Merge Replication to handle the Sync'ing portion and there seems to be some issues with those two together.

I am trying to understand what approach to use for the DAL. I originally had planned to use LINQ to SQL but I have read tidbits that state it doesn't work in a Sometimes Connected setting. I have therefore been trying to read and experiment with numerous solutions; SubSonic, NHibernate, Entity Framework. This is a relatively simple application and due to a "looming" verion 3 redesign this effort can be borderline "throwaway." The emphasis here is on getting a desktop version up and running ASAP.

What i am asking here is for anyone with any experience using any of these technology's(or one I didn't list) to lend me your hard earned wisdom. What is my best approach, in your opinion, for me to pursue. Any other insights on creating this kind of App? I am really struggling with the DAL portion of this program.

Thank you!

c# winforms linq data-access-layer

Share

Improve this question

Follow

asked Mar 9, 2009 at 19:38

community wiki
Refracted Paladin

Microsoft used to have some tech demos of this use case when .net 1.1 was still new. I think they called it disconnected record sets. Don't have time to Google at the moment but I'll check later on. – jason saldo Mar 9, 2009 at 19:55

4 Answers

Sorted by: Highest score (default)

1

If the stored procedures do what you want them to, I would have to say I'm dubious that you will get benefits by throwing them away and reimplementing them. Moreover, it shouldn't matter if you use stored procedures or LINQ to SQL style data access when it comes time to replicate your data back to the master database, so worrying about which DAL you use seems to be a red herring.

The tricky part about sometimes connected applications is coming up with a good conflict resolution system. My suggestions:

- Always use RowGuids as your primary keys to tables. Merge replication works best if you always have new records uniquely keyed.
- Realize that merge replication can only do so much: it is *great* for bringing new data in disparate systems together. It can even figure out one sided updates. It *can't* magically determine that your new record and my new record are *actually the same* nor can it really deal with changes on both sides without human intervention or priority rules.
- Because of this, you will need "matching" rules to resolve records that are claiming to be new, but actually aren't. Note that this is a fuzzy step: rarely can you rely on a unique key to actually be entered exactly the same on both sides and without error. This means giving weighted matches where *many* of your indicators are the same or similar.
- The user interface for resolving conflicts and matching up "new" records with the original needs to be easy to operate. I use something that looks similar to the classic three way merge that many

> source control systems use: Record A, Record B, Merged Record. They can default the Merged Record to A or B by clicking a header button, and can select each field by clicking against them as well. Finally, Merged Records fields are open for edit, because sometimes you need to take parts of the address (say) from A *and* B.

None of this should affect your data access layer in the slightest: this is all either lower level (merge replication, provided by the database itself) or higher level (conflict resolution, provided by your business rules for resolution) than your DAL.

Share Improve this answer Follow

edited Mar 18, 2009 at 20:26

community wiki
2 revs
Godeke

0

If you can install a db system locally, go for something you feel familiar with. The greatest problem I think will be the syncing and merging part. You must think of several possibilities: Changed something that someone else deleted on the server. Who does decide?

Never used the Sync framework myself, just read an article. But this may give you a solid foundation to built on. But each way you go with data access, the solution to the businesslogic will probably have a much wider impact...

answered Mar 9, 2009 at 19:44

community wiki
Sascha

0

There is a sample app called issueVision Microsoft put out back in 2004.
http://windowsclient.net/downloads/folders/starterkits/entry1268.aspx

Found link on old thread in joelonsoftware.com.
http://discuss.joelonsoftware.com/default.asp?joel.3.25830.10

Other ideas...
What about mobile broadband? A couple 3G cellular cards will work tomorrow and your app will need no changes sans large pages/graphics.

Excel spreadsheet used in the field. DTS or SSIS to import data into application. While a "better" solution is created.

Good luck!

answered Mar 10, 2009 at 0:33

community wiki
jason saldo

0

If by SP's you mean stored procedures... I'm not sure I understand your reasoning from trying to move away from them. Considering that they're fast, proven, and already written for you (ie. tested).

Surely, if you're making an app that will mimic the original, there are definite merits to keeping as much of the original (working) codebase as possible - the least of which is speed.

I'd try installing a local copy of the db, and then pushing all affected records since the last connected period to the master db when it does get connected.

Share Improve this answer Follow

answered Mar 17, 2009 at 15:14

community wiki
Steven Evers