Programming pattern using typed datasets in VS 2008

Asked 16 years, 1 month ago Modified 15 years, 11 months ago

Viewed 3k times

1

I'm currently doing the following to use typed datasets in vs2008:

Right click on "app_code" add new dataset, name it tableDS.

Open tableDS, right click, add "table adapter"

In the wizard, choose a pre defined connection string, "use SQL statements"

select * from tablename and next + next to finish. (I generate one table adapter for each table in my DB)

In my code I do the following to get a row of data when I only need one:

cpcDS.tbl_cpcRow tr = (cpcDS.tbl_cpcRow)(new cpcDSTableAdapters.tbl_cpcTableAdapter()).GetData().Select("cpcID = " + cpcID)[0];

I believe this will get the entire table from the database and to the filtering in dotnet (ie not optimal), is there any way I can get the tableadapter to filer the result set on the

database instead (IE what I want to is send select * from tbl_cpc where cpcID = 1 to the database)

And as a side note, I think this is a fairly ok design pattern for getting data from a database in vs2008. It's fairly easy to code with, read and mantain. But I would like to know it there are any other design patterns that is better out there? I use the datasets for read/update/insert and delete.

c# sql visual-studio-2008 .net-2.0 dataset

Share

Improve this question

Follow

edited Dec 19, 2008 at 11:22

AnthonyWJones

189k ● 35 ● 235 ● 307

asked Nov 17, 2008 at 10:44

devzero

2,670 ● 5 ● 38 ● 59

3 Answers

Sorted by: Highest score (default)

1

A bit of a shift, but you ask about different patterns - how about LINQ? Since you are using VS2008, it is possible (although not guaranteed) that you might also be able to use .NET 3.5.

A LINQ-to-SQL data-context provides much more managed access to data (filtered, etc). Is this an option?

I'm not sure I'd go "Entity Framework" at the moment, though (see here).

Edit per request:

to get a row from the data-context, you simply need to specify the "predicate" - in this case, a primary key match:

```
int id = ... // the primary key we want to look for
using(var ctx = new MydataContext()) {
   SomeType record = ctx.SomeTable.Single(x => x.SomeC
   //... etc

   // ctx.SubmitChanges(); // to commit any updates
}
```

The use of Single above is deliberate - this particular usage [Single(predicate)] allows the data-context to make full use of local in-memory data - i.e. if the predicate is just on the primary key columns, it might not have to touch the database at all if the data-context has already seen that record.

However, LINQ is very flexible; you can also use "query syntax" - for example, a slightly different (list) query:

```
    var myOrders = from row in ctx.Orders
                   where row.CustomerID = id && row.Is
                   orderby row.OrderDate
                   select row;
```

etc

Share Improve this answer

Follow

edited May 23, 2017 at 11:49

Community Bot

1 • 1

answered Nov 17, 2008 at 11:03

Marc Gravell

1.1m • 273 • 2.6k • 3k

LINQ-to-SQL data context is an option. Could you give me some example code doing the same as above? (Ie fetch a row of data from the table tbl_cpc ?) – devzero Nov 17, 2008 at 11:45

1

There is two potential problem with using typed datasets,

one is testability. It's fairly hard work to set up the objects you want to use in a unit test when using typed datasets.

The other is maintainability. Using typed datasets is typically a symptom of a deeper problem, I'm guessing that all you business rules live outside the datasets, and a fair few of them take datasets as input and outputs some aggregated values based on them. This leads to business logic leaking all over the place, and though it will all be honky-dory the first 6 months, it will start to bite you after a while. Such a use of DataSets are fundamentally non-object oriented

That being said, it's perfectly possible to have a sensible architecture using datasets, but it doesn't come naturally. An ORM will be harder to set up initially, but will lend itself nicely to writing maintainable and testable code, so you

don't have to look back on the mess you made 6 months from now.

Share Improve this answer

Follow

answered Nov 17, 2008 at 11:17

AndreasKnudsen

3,481 ● 5 ● 31 ● 33

Since you obviously don't like datasets, how would you suggest I get the data from the database and into the code? – devzero Nov 17, 2008 at 11:48

Very few object technologies are simpler to set up objects than DataSets for testing. 1) fill with data (by hand or by query), 2) save to xml, 3) re-fill from that saved xml for your tests (one example). Also, with their partial classes, it's silly to say DataSet is "non-object oriented". – Mark A Johnson Jan 9, 2009 at 21:03

0

You can add a query with a where clause to the tableadapter for the table you're interested in.

LINQ is nice, but it's really just shortcut syntax for what the OP is already doing.

Typed Datasets make perfect sense unless your data model is very complex. Then writing your own ORM would be the best choice. I'm a little confused as to why Andreas thinks typed datasets are hard to maintain. The only annoying thing about them is that the insert, update, and delete commands are removed whenever the select command is changed.

Also, the speed advantage of creating a typed dataset versus your own ORM lets you focus on the app itself and not the data access code.

Share Improve this answer Follow

answered Jan 9, 2009 at 19:44

Spivonious