Creating a class to use for populating drop-down lists, grids, etc., in C#

Asked 15 years, 10 months ago Modified 15 years, 10 months ago Viewed 1k times

0

Please keep in mind that I'm new to C# and OOP so I apologize in advance if this seems like an easy question to some. I'm going back through my code and looking for ways to objectify repetitive code and create a class for it so that I can simply reuse the class. That being said, I'm not looking to learn NHibernate or any other ORM just yet. I'm not even looking to learn LINQ. I want to hack through this to learn.

Basically I use the same bit of code to access my database and populate a drop-down list with the values that I get. An example:

```
   protected void LoadSchools()
    {
        SqlDataReader reader;

        var connectionString =
ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;
        var conn = new SqlConnection(connectionString);
        var comm = new SqlCommand("SELECT * FROM [Schools] ORDER BY
[SchoolName] ASC", conn);

        try
        {
            conn.Open();
            reader = comm.ExecuteReader();
            cmbEditSchool.DataSource = reader;
            cmbEditSchool.DataBind();


            cmbEditSchool.Text = "Please select an existing school to edit...";
            if (reader != null) reader.Close();
        }

        finally
        {
            conn.Dispose();
        }
    }
```

I use this same bit of code, over and over again throughout my program, on different pages. Most often, I'm populating a drop-down list or combo box, but sometimes I will populate a gridview, only slightly altering the query.

My question is how can I create a class that will allow me to call a stored procedure, instead of manually using queries like my example, and populate my different controls? Is it possible to do with only 1 method? I only need to start with

selects. I've been reading up on IEnumerable which seems like the appropriate interface to use, but how do I use it?

Edited to add:

I marked Rorschach's answer as THE answer because s/he addressed my IEnumerable question. I also understand the need for a proper DAL and perhaps BLL layer. What I was trying to get to was that. I can build a DAL using datasets and table adapters which, in the end, gets me a strongly typed dataset. However, I feel a bit removed from the code. I was after a straight-forward way of building the DAL myself starting with the code that I gave above. Perhaps I'm not wording my question, or what I'm after, correctly.

At any rate, Rorschach came closest to answering my actual question. Thanks.

c# oop ienumerable

Share
Improve this question
Follow

edited Feb 10, 2009 at 20:39

asked Feb 10, 2009 at 18:33

GregD
7,000 ● 5 ● 36 ● 61

3 Answers

Sorted by: Highest score (default)

2

You can create a class that lets you call stored procedures (this is known as a Data Access Component (DAC) class, which is usually referenced by a Business Component (BC) class, but it is outside the scope of your question).

There are a few objects you will want to use in this new class:
Microsoft.Practices.EnterpriseLibrary.Data.Database
Microsoft.EnterpriseLibrary.Data.DatabaseFactory
System.Data.Common.DBCommand

The DAC class will look similar to what you have:

```
public class DataAccess
{
    public DataAccess()
    {
    }
    public System.Collections.IEnumerable GetSchoolData()
    {
        string connectionString =
ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;
        Database db = DatabaseFactory.CreateDatabase(connectionString);
        string sqlCommand = "GetSchoolData";
```

```
        DbCommand comm = db.GetStoredProcCommand(sqlCommand);
        //db.AddInParameter(comm, "SchoolId", DbType.Int32); // this is in case
you want to add parameters to your stored procedure

        return db.ExecuteDataSet(comm);
    }
}
```

And your page code will look like this:

```
public class SchoolPage : Page
{
  public void Page_Init(object sender, EventArgs e)
  {
    DataAccess dac = new  DataAccess();
    cmbEditSchool.DataSource = dac.GetSchoolData();
    cmbEditSchool.DataBind();
  }
}
```

Note that this is just to help you learn how to do this. It is not a good approach to development because you are opening up your Data Access Layer to the outside world (which is bad).

Share Improve this answer Follow

answered Feb 10, 2009 at 19:06

Ryan Abbott

5,397 ● 7 ● 33 ● 34

2

This is a down and dirty method of reducing your code duplication. It's not really the right way to go about setting up a Data Access Layer (DAL) and a Business Logic Layer (BLL), which I'd suggest learning about instead.

```
protected void FillFromDatabase( string sql, BaseDataBoundControl dataControl)
{
SqlDataReader reader = null;

var connectionString =
ConfigurationManager.ConnectionStrings["MyConnectionString"].ConnectionString;
var conn = new SqlConnection( connectionString );
var comm = new SqlCommand( sql, conn );

try
{
    conn.Open();
    reader = comm.ExecuteReader();
    dataControl.DataSource = reader;
    dataControl.DataBind();
}

finally
{
    if( reader != null )
```

```
        reader.Dispose();

    conn.Dispose();
}
}
```

then you could call it like

```
const string sql = "SELECT * FROM [Schools] ORDER BY [SchoolName] ASC";
FillFromDatabase( sql, cmbEditSchool );
```

Share

Improve this answer

Follow

answered Feb 10, 2009 at 18:51

community wiki
Greg

Greg beat me to the punch almost verbatum, so I'll just reiterate what he's saying. What you're trying to do is a very very bad idea. You're looking to do something that you're going to reuse everywhere, but that's just going to propagate a bad non-OOP solution to many places in your solution. – Michael Meadows Feb 10, 2009 at 18:54

I've made this community wiki, so feel free to improve the answer as you see fit. – Greg Feb 10, 2009 at 18:55

Why is this a bad idea? I thought I was headed toward building a DAL & subsequently a BLL and getting code reuse out of the same class. – GregD Feb 10, 2009 at 18:58

First, you're marrying your implementation to a technology (in this case a database). Second, if you want to switch to a disconnected solution in the future, you will have to rewrite vast swaths of code. Third, this is not an OO solution, it's procedural. – Michael Meadows Feb 10, 2009 at 19:07

Fourth, it does not allow you to write automated unit tests. Fifth, any change in the database must propagate to the UI in sync, and vice versa. Over time, entropy will cause all code to get worse, refactoring this code is more difficult because there's no indirection.
– Michael Meadows Feb 10, 2009 at 19:08

0

You should take a look at ADO.NET and use things like DataSets. Linq-to-SQL would probably also be useful for you. But you probably won't get away with just one method call to replace all your data operations I'm afraid.

Share Improve this answer Follow

answered Feb 10, 2009 at 18:48

netterdotter
131 ● 3 ● 12

I appreciate the info, however, I don't necessarily want to use a dataset. I don't have a need for disconnected data and I really want to learn how to do this from the ground up...so to

speak. I can drop a SqlDataSource control on my page and be done with it. That's not helping me learn... – GregD Feb 10, 2009 at 18:51

@GregD, I'd argue that learning a bad practice isn't the first step to "ground up" learning. The aversion to binding directly to a data reader goes much deeper than just a "preference," it's a bad practice for many reasons. Binding data directly to forms is a loosing formula all around. – Michael Meadows Feb 10, 2009 at 18:59

I guess I'm not following you Michael. I don't have an aversion to binding directly to a datareader. I was simply trying to reuse code and was asking for a way to create a class (beginnings of a DAL) that would have a "universal" datareader in it and I could use the class instead. – GregD Feb 10, 2009 at 19:04

@GregD, by aversion, I'm referring to the responses to your question, not your own. If you do it, you'll be paying for that decision for the entire application life cycle. Because there's no indirection, you'll be less able to fix it by going to a well established data access pattern in the future. – Michael Meadows Feb 10, 2009 at 19:14