

Use Stack Overflow for Teams at work to find answers in a private and secure environment. Get your first 10 users free. [Sign up.](#)

# Where do you put SQL Statements in your c# projects?

Asked 10 years, 6 months ago   Active 2 years, 11 months ago   Viewed 19k times



23



8

I will likely be responsible for porting a vb6 application to c#. This application is a windows app that interacts with an access db. The data access is encapsulated in basic business objects. One class for one table basically. The existing vb6 business objects read and write to the DB via DAO. I have written DALs and ORMs a few times before but they all targeted SQL Server only. This one will need to target access and sql server. In previous projects, I would place the SQL strings in the private parts of the business object and maybe move the redundant sql code like connecting, creating command, in into a common base class to reduce the code.

This time, i'm thinking about writing the SQL strings into a .settings file or some other key/value type text file. I would then write a sql utility to edit this file and allow me to run and test the parameterized queries. These queries would be referenced by name in the business object instead of embedding the sql into code.

I know a standard approach is to create a DAL for each targeted database and have the configuration state which DAL to use. I really don't want to create the two DAL classes for each database. It seems like it would be less code if I just referenced the correct query by keyname and have the proper type of connection.

So, are you guys doing things like this? How would or have you approached this problem? What works best for you?

Thanks!

c#

sql

winforms

orm

data-access-layer

asked Feb 23 '09 at 4:49



Steve

1,525

4

17

31

## 9 Answers

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



"build action". That way, the SQL is included in your resulting assembly, and can be retrieved from it at runtime using the



ResourceManifestStream of the .NET framework:



```
private string LoadSQLStatement(string statementName)
{
    string sqlStatement = string.Empty;

    string namespacePart = "ConsoleApplication1";
    string resourceName = namespacePart + "." + statementName;

    using(Stream stm =
Assembly.GetExecutingAssembly().GetManifestResourceStream(resourceName))
    {
        if (stm != null)
        {
            sqlStatement = new StreamReader(stm).ReadToEnd();
        }
    }

    return sqlStatement;
}
```

You need to replace "ConsoleApplication1" with your actual namespace, in which the sql statement files reside. You need to reference them by means of the fully qualified name. Then you can load your SQL statement with this line:

```
string mySQLStatement = LoadSQLStatement("MySQLStatement.sql");
```


This however makes the queries rather "static", e.g. you cannot configure and change them at runtime - they're baked right into the compiled binary bits. But on the other hand, in VS, you have a nice clean separation between your C# program code, and the SQL statements.

If you need to be able to possibly tweak and change them at runtime, I'd put them into a single SQL table which contains e.g. a keyword and the actual SQL query as fields. You can then retrieve them as needed, and execute them. Since they're in the database table, you can also change, fix, amend them at will - even at runtime - without having to re-deploy your whole app.

Marc

answered Feb 23 '09 at 6:28

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

- 
- 2 I've used this approach and I like it. Unlike most people, I don't agree with being able to change code on the fly at runtime, even simple SQL statements, so the fact that it's compiled with the assembly is a plus for me. – [Chris](#) Mar 8 '09 at 16:25
- 
- 1 Glad to hear this approach is being used and welcomed by others! :-) And I agree - sometimes baking stuff into your compiled bits is a plus. – [marc\\_s](#) Mar 8 '09 at 17:34
- 
- 3 Instead of namespacePart, use this.GetType().Namespace for current, dynamic version, and typeof(Program).Namespace for static. Of course when you put it into Program class as static. Don't forget, when your SQL file is in SQL directory, refer to it by "SQL.StatementName.sql" – [Harry](#) Mar 30 '12 at 7:49 
- 
- I like the dynamic approach, using a SQL table. Just one thing that worries me is source safe. In the first instance, the queries would be part of the project, and thus source safe, whereas a manual save to source safe is needed in the SQL Table instance. Still handy though. – [Adriaan Stander](#) Sep 19 '12 at 12:39
- 



10



When I really need it, I put the queries into individual \*.sql files, then include them into Resources.resx. There is a 'Files' section in it, which allows you to include Embedded Resource files.

After that, I can use generated Resources.MyQuery property which both guarantees that resource exists and saves me from writing a custom resource load method.

answered Feb 23 '09 at 7:52



[Andrey Shchekin](#)

12.6k 12 81 138

---

I also like this solution too. The generated properties are handy. – [Steve](#) Feb 23 '09 at 14:54

---



2



**LINQ to DataSet** sounds like the way to go for you.

If you haven't used the .NET 3.5 before / LINQ then you're in for a treat. LINQ will save you writing your raw sql in string literals and provide you with a more logical way to creating queries.

Anyway, check this link out for using LINQ on Access databases - <http://msdn.microsoft.com/en-us/library/bb386977.aspx>

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

2

If i'd had to create application for both SQL and Access, I'd use some IDAL interface, DALCommon with common functionality implementation and separate DALSql and DALAccess, inherited from DALCommon, with some specific stuff, like exceptions, transactions handling, security etc.

I used to keep stored procedure names or queries in resource files.

answered Feb 23 '09 at 13:22



[Maksym Gontar](#)

21.4k 9 74 112

1

I'll tell where I won't put it ever, something I saw done in some code I inherited. It was in Java, but applies to any language

- A base class that declared protected static member variables for for SQL statements, init'd to null, with a get method that returns individual SQL statements
- A sub class for each supported database server, with an init method that assigns to the base class member variables
- Several DA classes that use the base class method to retrieve SQL statements
- The application start-up class with the responsibility to create the correct sub-class object and call its init method

I will also not go into explaining why I will not do this ever :-)

answered Feb 23 '09 at 5:23



[Miserable Variable](#)

24k 11 59 112

2 OO purist language causes OO nightmares like this! funny! – [Steve](#) Feb 23 '09 at 16:57

1

One method we used is to have a class that would connect to the DB and methods to call procedures and in the method parameter you would provide the procedure name. so all the SQL code is in the procedure. we would use overloads for the different return types

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).

```
XMLDataDocument runProcedure(string procedureName);  
int runProcedure(string procedureName);  
  
//etc....  
}
```

answered Feb 23 '09 at 5:36



Mike

1,995

19

32

---

You can't have overloads which differ only by return type - it will give a compiler error CS0111 – [MattDavey](#) Aug 21 '17 at 16:11

---



0

Sometimes, like with custom reporting apps, you really need to embrace the impedance mismatch, and give special importance to the SQL. In these cases I recommend the following: For each module that contains SQL strings, create a single static "SQL" class to hold them all. Some of the SQL strings will likely require parameters, so be consistent and put each string behind it's own static method.



I only do this for the occasional custom reporting app, but it always works out great and feels refreshing and liberating. And it's quite nice to come back months later to make an enhancement, and find all of the SQL waiting for you in a single SQL.cs file. Just by reading that one file, it all comes back, and often this is the only file that needs to be changed.

I don't see a need in these cases for hiding the SQL in resources or elsewhere. When SQL is important, then it's important. Interestingly, more and more developers are now freely mixing SQL with C#, including I believe this site, because essentially, that's what LINQ is.

Finally, as always, make sure you are not susceptible to SQL injection attacks. Especially if user input is involved, make sure you are using some kind of parameterization and that you are not using string concatenation.

edited Mar 8 '09 at 16:11

answered Mar 8 '09 at 15:38



Mike

1,200

13

28



0

Embedding solutions shown above may not work if SQL Query has a "where" clause like , but for the same Query the next run needs PropertyID='113' as the PropertyID is read-in.

By using our site, you acknowledge that you have read and understand our [Cookie Policy](#), [Privacy Policy](#), and our [Terms of Service](#).



[Glad you asked! Put your sql in a QueryFirst .sql template.](#)

0



It's automatically compiled into your app as an embedded resource, but you don't care. You just write it, in a real sql window, connected to your DB, with syntax validation and intellisense for tables and columns, then use it, via the generated `Execute()` methods, with intellisense for your inputs and results.

disclaimer : I wrote QueryFirst.

answered Oct 6 '16 at 9:57



[bbsimonbb](#)

12.7k

6

41

57