

Re: Programming Practices

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2004-04/4979.html>

From: William Ryan eMVP (*bill_at_NoSp4m.devbuzz.com*)

Date: 04/21/04

Date: Wed, 21 Apr 2004 02:26:20 -0400

Hi Ed_P:

"Ed_P" <anonymous@discussions.microsoft.com> wrote in message
news:010D2835-50DA-4384-946D-62A3B226FF0F@microsoft.com...

> *Hello,*

>

> *I just wanted to get the opinions of those of you who have experience*
developing C# applications and programming in general. I currently am
learning the basics of programming (choosing C# as the language of choice)
and have developed simple applications to work with Access/SQL databases. I
have accomplished all tasks that I wanted to with the small but simple
applications...learned alot about the syntax and logic used in C# an
programming in general. However I feel as though I my applications, as
trivial as they are could be better.

No matter how long you program, you should always have this feeling about
your code. The worst thing that can happen to a programmer is if he/she
starts falling in love with their code, become convinced that their way is
the only way or the best way and quits question what they do and why they do
it. If there's ever a profession that you should constantly get better at,
it's this one.

>

> *My question to all who wish to provide some feedback on this topic is what*
can I do to help me become more proficient in programming (no in C# but in
any OOP language). Things that I would like to do create an application
that can have features or bug fixed later on by using service packs or
hotfixes. With my current applications I only have one EXE file that will
launch the app, I would like to see if I could maybe break this up with
different dll's to be used by the application to make it more easy to manage
and upgrade for future use. Making the application use less memory (for the
small c# applications), etc.

>

> *Can anyone give me any resources on the net or books that I can use. I*
have thought about looking over some of these open source products that you
can join to develop the software, but this seems out of my league right now.
In a nutshell, anything that anyone in here can provide me with to get me
more experience in programming (using C# and .NET specifically) will be

great!

There are soo many it'd actually be hard to go wrong. Let me just start with a few things though. You often hear people talking about breaking up code into 'layers', you hear about the presentation layer, the data access layer etc. By and large, the first thing you want to do is seperate your UI from your business logic. If nothing else, make sure you do this. I'll explain in a second. now, people often talk about business logic as this monolithic thing, but depending on how it's built, this may include your data access code, part of your data access code, your communications layer if applicable etc. Then you can break it up further by using the database backend. I have an app that's one of the better pieces of code that I've built. It's scaled well and user requests are always coming and changing.. originallly people might have loved a feature but now that they have it they take it for granted and want more. This particular app is a web app, so if I had the Data access logic in the appliccation or behind the ui, every time I changed a query, I'd have to rebuild the project and redeploy it. This would cause someone to have to load it on the web server at a minimum so if I did this a lot, it would be a real inconvenience. I could easily ask for a password and account, but in today's environment of increased security threats, many sysadmins don't want any unnecessary accounts out there and they don't want to grant permissions that aren't necessary.

I implemented this with a data access layer, a business logic layer and the ui. The data access layer calls stored procedures, and those stored procedure names are stored in a db related to user accounts. So, all my data access layer does is find out what proc it needs and then calls it. More times than I can count users have wanted something cosmetic changed that would otherwise require a recompile. However, while on the phone I can change the logic in the stored proc, hit save, have them refersh their page (which causes a new page view) and right before their eyes their changes are in effect. I'm still involved in the process, but due to security concerns and govt regs, we are limited in how much we can do without actually requesting a faxed permission first. Think about it if you were a user.. you want something changed. In one case i'd have to make the change, wait for someone from networking to come over and copy it, and then you could have your changes. Optimistically it'd take 10 minutes but it could take a lot more (if everyone in networking was busy restoring a drive controller that just crashed or was at lunch). so 10+ minutes compared to real time changes while you are on the phone.

This is just one example/benefit of seperating your logic. So the first thing you need to do is code everything with the assumption that it will change. Hard code nothing. Find a mechanism, database, web service whatever that you can store this stuff in so you can change it without having to rebuild the app. Tell yourself that each time you rebuild app, it's bad b/c you are taking a stable system and possibly injecting instability into it.

The same logic holds for .dlls'. If my UI is sound, and I need to change my data access layer to point somewhere else, if I recompile the whole app, I

may have introduced some other subtle bug (that's not to say that you can't still screw things up, but you are limiting the scope of what you can break). So break things up as small as logically sensible. Think about real life objects. Your car for instance. If you had to have the car rebuilt and sent back to the car factory to change a flat tire, you'd be mad. So by building it in pieces that resemble nouns, you simulate this. If your car object had 4 tire objects, each tire had a rim object etc, you could easily change one thing with minimal exposure to your other object. What would be the risk to your seat objects if you reimplemented the Rim object? very little. So when you are building your objects, always ask "will this change easily if I need it to" and does this object make logical sense?

You'll also need to understand events, methods and properties, and once again, model them with the same thought of real world objects. If you had a human object, would breathe be an event, method or property? This is pretty easy but other things may be more complex. You'll always want to ensure that you have as very little code behind your forms... if you have a lot of stuff there, you know you are doing something wrong.

Then I'd get a few books on OOP, look on amazon and find some that have many good reviews. OOP is a mature subject so there will be a good amount of feedback. Then get a book or two on .NET (or 20) and OOP and .NET in particular. You'll probably want at least one (like David Sceppa's ADO.NET core reference or Bill Vaughn's ADO ADO.NET Best practices) data access. You'll probably want a book on the backend db you are using too.

Most every .NET book will have a discussion of deployment, and check out Zero Touch deployment on google. This will help you out immensely.

It's a little late but I'll post a list of books I really liked.

And by all means, ask people questions and look at code .. look at examples on web sites and see what you like and dislike about it. Over time your style will develop and you'll probably use techniques from many different people.

HTH,

Bill

>

> *Thanks In Advance!*

>

> *Ed P.*