Starting out, any suggestions?

Asked 15 years, 8 months ago Modified 15 years, 8 months ago

Viewed 181 times

0

I have started working in C# for almost a few months and I am looking for something more challenging and interesting. I use a media player called media monkey that supports custom vb scripts, well I made one that writes a file to a dir that has the current song playing, and is updated every time a new song is playing by rewriting what was there before.

Now I want to add this information to a database and keep a record of this and possibly add the information on my home page. I know I can hack a way for it to work, but I want to know what would be the "professional way" of doing things.

I came up with the following and got stuck. I would need an ODBC driver to connect to a database which seems messy, would a web service work? How would that work? Can a VbScript call a dll file to call upon a web service to modify data on a seperate server? Is that safe to do?

vbscript

edited Apr 1, 2009 at 5:07

Share

Improve this question

Follow

John Sheehan
78.1k ● 30 ● 161 ● 194

asked Apr 1, 2009 at 5:04

1 Answer

Sorted by: Highest score (default)

1

Many professional C# apps are n-tier. In your case, you would probably layer it like this:

On the server:
-Database Store
-Database Access/Business layer(sometimes two distinct components, depending on how complex the app is)
-Web Service

On the client:
-Web Service Client
-Any other layers to support client functionality.

So the Database Store would be something like some tables in an Oracle or Microsoft SQL Server, and would on your server.

Database Access/Business layer would be your code that retrieves and stores data to/from your database. It might also contain business objects, which are basically classes that have properties representing your data from your database. The benefit of the data access layer is that sometimes reading and writing to a database can require specialized code, and you don't want that code sprinkled throuought your application. So instead you can call functions in your data access layer that loads needed data into objects, so the rest of your application is just interacting with a regular old .NET object/class. These are called POCOs, which stands for something like Plan Old CLR Object. There are lots of variations on this of course, as people have taken different approaches to the problem of isaloting database access. Also it serves the purpose of minimizing breaking changes whenever the database changes. Since the database access logic is not sprinkled throughout the app, then there are fewer places that need to be updated if the database changes (such as adding new columns to a table or changing a name).

Sometimes the business layer will be it's own layer, and would contain most of the "logic" of the application. It would sit between the data access and web service layers. Using concepts from Service Oriented Architecture (SOA), you might have an authentication service, and a web request handling service. These services are a lot like a class that is always instantiated, there waiting to process requests. Your web request handling service would take a request, and maybe first call into the authentication service to verify credentials before honoring the request. SOA is one of those things I think should be used only when appropriate. It some cases just using Object Oriented techniques will give you the same benefits. Not always though. SOA, when done right, is more scalable, so it really depends on whether SOA offers you additional benefits that you need.

The Webservice would be responsible for receiving requests from the web, parsing/interpreting them, and acting on those requests by making calls into your business layer to update or retrieve data.

So the concept here would be that you could have many users of your service who publish their song updates through your service.

Your client would have a "web service client" layer which would be responible for formatting requests into messages, sending them to the web service, and retrieving messages from the web service. You would put very little application "logic" in your web service layer.

Now all this is probably overkill and inefficient for what you are wanting to do since you just want something for yourself, but it's the basic anatomy of a lot of webservice applications and would be a good learning exercise. The whole purpose of the layers is decoupling and simplicity. While more layers/components makes the application overall more complex, it means each component is simpler. This means it's easier to wrap your head around problems when you are only dealing with one component which interacts with only a couple other components(the sourounding layer). So there is a careful balance between few components and many components. Too few and they become monolithic and difficult to manage. Too many, and they become intertwined in complex ways. I have heard it said something along the lines of "If a class is getting too big and too complex, then split it up into a few more classes". In essence, don't start subdividing stuff for the heck of it just because it sounds like the right thing to do. Evaluate how complex your component is going to be before deciding if you want to split it up. Sometimes for simple cases your have a layer serving more than one purpose, for the sake of getting it done faster and making the overall design simpler. The point is, apply these concepts where appropriate. You will learn what is appropriate with experience, and you obviously understand that you can learn the most by "doing".

"Can vbscript call a COM component?" You can compile .NET DLLs with COM support. Many older things can call COM dlls.

I googled: vbscript dll and got this: [VB Script and DLLs]

"Is that safe to do?" Your webservice will be where you would be most concerned with security. It's safe only if you design with security in mind and don't screw up. We all screw up sometimes though, which means there is no guarantee of it being perfectly secure.

Share Improve this answer Follow

answered Apr 1, 2009 at 5:54

AaronLS

38.3k ● 23 ● 148 ● 214