How can I convince skeptical colleagues about proper namespaces in .Net?

Asked 16 years ago Modified 10 years, 3 months ago Viewed 717 times

5

My team is working on a conversion project to convert one product (but with many facets) from VB6 to .Net (we have over ~300k LOC). Before I got on board, the decision was made that regardless of the location of the assembly, or folder structure, all classes / structs will be in one namespace:

.

They even go as far as changing auto-generated application settings designer code, resource designer code, and such to force uniformity. How do I convince them that namespace use is good? What is the proper use of namespace and what are the pros and cons? I guess I am having a tough time understanding why my colleagues would go through so much hassle to save a few using lines. Any external, reputable references to support your argument would be much appreciated. Please help!

.net namespaces

Share

Improve this question

Follow

edited Aug 26, 2014 at 15:38

Robert Harvey

181k ● 48 ● 346 ● 512

asked Dec 20, 2008 at 3:57

Echiban

847 ● 2 ● 11 ● 21

they could just change the default namespace setting in the project properties instead... – Steven A. Lowe Dec 20, 2008 at 6:21

That doesn't resolve the issue entirely since they create subfolders, since the default namespace in those folders is: <default_namespace>.<folder_name>. Also, resources and settings files are automatically in designated folders, which will have a different namespace. – Echiban Dec 20, 2008 at 10:25

11 Answers

Sorted by: Highest score (default)

Good answers, already

7

I read a lot of good answers (the drive folder's image is a good one, for example, as is the object/concept separation/organization).

Still, some other, less "technical" arguments could help, not to explain why namespaces are better than the alternative, but to unearth the bad reasons namespaces are not used.

Argument by Authority

Every C#, VB.NET, Java, C++, whatever developer worth its salt on this planet acknowledge namespaces/packages are a good thing. Hey, they are even considering it in Mozilla's JavaScript.

I guess that your question on Stack Overflow has the "Authority Argument" touch... ^_^

Old Habits die hard

I believe you are falling in a case of "Old habits".

While I have no direct experience about VB.NET against VB6 old habits, I do for C++ against stripped-down C-like, C++.

Half the work...

When dealing with legacy habits, half the work is discovering they already use namespaces, without acknowledging it.

For example (this is most true for C++ devs colliding with "old fashion", C-like code, but it is viable everywhere, I guess), do they use prefixes for their symbols.

Do they have:

- *Account*User
- *Account*Price
- *Account*Id
- *Login*User
- *Login*Password

classes/symbols, and explain that the "User" of Account is not the same than the "User" of login, and thus, the prefix to differentiate them?

The "pattern" go right through the ceiling when they prefix their symbols by module (which is certainly the case for your code, if it is large enough):

Do you have a MyUtil DLL, as well as MyKernel DLL and a MyGui DLL ? Thus, do you have:

- *MyUtil*_Something
- *MyUtil*_SomethingElse
- *MyKernel*_YetAnotherObject
- *MyKernel*_AnotherThing
- *MyGui*_SomeGui

classes/symbols?

If yes, then they should acknowledge they are already using namespaces.

If not, then either your codebase is too small (but your "9 projects" tell me it's not), or they are regularly problems of

name collision... Which can be resolved either by the prefixing explained above, or even better, by namespaces.

The other half...

The other half of the work is making them explain why their "namespaces" are better than the one built-in in the language (*this is the moment you should keep from banging your head against the walls in frustration because of lethal exposure braindead reasoning*).

Again, you can use the Argument by Authority thing (*What? You do believe know better than the Top Gun engineers from Microsoft?*), but in this case, you should give examples of why the .NET (or whatever) namespaces are better than the one they use (or don't use).

For VB.NET/.NET, there already more than enough good arguments, and I won't go on the ones behind C++ namespaces because it would be out of topic.

But at the very least, using built-in namespaces makes the prefix mentioned above optional. Quoting an example from internet (I'm not very knowledgeable of VB.NET):

```
Imports MyWholeProject

Class HelloWorld

   Public Sub Main()
      Dim o As MyModuleAAA_MyObject = New MyModuleAAA_
```

```
        Dim t As MyModuleBBB_MyThing = New MyModuleBBB_M
        Dim g As MyModuleCCC_MyGizmo = New MyModuleCCC_M
        REM etc.
    End Sub

End Class
```

Is quite more verbose than the namespace enabled:

```
Imports MyWholeProject.MyModuleAAA
Imports MyWholeProject.MyModuleBBB
Imports MyWholeProject.MyModuleCCC

Class HelloWorld

    Public Sub Main()
        Dim o As MyObject = New MyObject
        Dim t As MyThing = New MyThing
        Dim g As MyGizmo = New MyGizmo
        REM etc.
    End Sub

End Class
```

And verbosity lead to more obscure code, and thus bugs. Of course, you could replace MyModuleAAA by something less long, like MMAAA... But then, it leads to more obscure code, etc. etc..

More uses of namespaces on .NET can be found by Googling, like: http://www.vbdotnetheaven.com/UploadFile/ggaganesh/NamespacesInVbDotNet04202005005133AM/NamespacesInVbDotNet.aspx

Project/Career related reasons

Those reasons are more relatived to project/career management...

Maintenance is not optional

If your application is an old dying cow, where changing a label or the content of a combo is worth 2 days of work (it *does* happen) then there is no point, perhaps, to an extra-work of refactoring. In the other hand, if the refactoring is automated, then you should do it anyway.

But if your application is supposed to be used in the future, then sabotaging it to keep it in the past is a bad idea. You'll pay for it, someday. Each hour supposed to be won by avoiding maintenance will be paid twice or tenfold the day it will become unavoidable (you know, the day when you won't have time to do it properly because it is urgent...)

Keeping oneself up-to-date is not optional

This argument is for engineers, who have a career, and must keep their job: Like applications, software engineers can become obsolete.

And it means that the obsolete developer, when applying to another job, will lose, too. Because, hey, who wants to pay for someone who just don't get something as simple as namespaces?

Experience is a good thing when it is coupled with knowledge of current state-of-the-art. If not, then it's only dinosaur-old cynical and useless rambling about "*how it was better*(read: simpler)*, in my time, when we did not need all those gizmos like objects*".

Keeping oneself up-to-date is not optional (manager side)

By refusing to adapt to new features of the language (usually because "*we know better, you know?*"), engineers are just putting on concrete shoes while the competition is probably putting running shoes. This mean the application produced by an obsolete team will eventually lose. Period.

In a similar fashion, the team will hire new engineers, who probably know about the features, and who will become surprised, and then frustrated about the fact they are just programing like their grand-mothers did decades ago. The obsolete team won't keep new engineers very long (at least, the ones worth their pay).

Note that working on an obsolete product make it easier for top-level managers to decide it would get easier to

rewrite it from scratch... And possibly somewhere else, with less-paid engineers and managers...

Share Improve this answer

Follow

answered Dec 20, 2008 at 13:42

paercebal

83.2k ● 38 ● 134 ● 160

@Greg Dean: I know... This is *my* flaw (among others), as far as my own managers are concerned. Anyway, the titles I used are there to cut the text into smaller parts. For example, if your interested in the "Project/Career related reasons", then you can avoid the whole text... ^_^ ... – paercebal Dec 31, 2008 at 16:57

@Greg Dean: I'll note, through, that your "What do they think namespaces are for? Why do they think the BCL uses them?" is too short, and between our comments, they have av average interesting size... ^_^ ... – paercebal Dec 31, 2008 at 17:00

5

The proper use of namespaces is to provide an orderly structure of logically grouped classes, interfaces, enums etc. It's like putting all your files in a single folder... or worse, putting them all in the root of C: drive...

Far be it for me to point out that your team is being extremely shortsighted, it's not really my place to start questioning other teams policies and procedures - but if I were in your position, I'd stick to my guns.

Share Improve this answer

Follow

answered Dec 20, 2008 at 4:22

BenAlabaster

39.8k ● 22 ● 114 ● 151

Hopefully your team understands object-oriented design principles. You group related information into classes that have operations that work on that information. Namespaces are (or should be) similar. You use namespaces to group related classes together and together they provide operations to work on the collective information of the namespace.

This helps with code intelligibility. Grouping your data layer classes into a single namespace identifies them as related, and related in a way to provide the functionality indicated by the namespace [As an aside, this is why you would need to provide meaningful names for your namespaces as well as your classes]. If you group all of your classes in the same namespace, then it all becomes a big ball of mud. You've lost the semantic clues to the organization of the code and have to dig through it to find out which classes work together to provide the required functionality.

The point is: it's not merely a matter of personal preference. In the same way that you can't expect to pour all of your sentences in to a single paragraph, but rather have to organize them into several paragraphs, in which the sentences are related; you need to take your classes and organize them into namespaces. Breaking your writing up into paragraphs of related sentences provides clarity and meaning, breaking your classes into namespaces improves the clarity of your code. As the

number of classes grows, this only becomes more and more important.

Share Improve this answer Follow

edited Dec 20, 2008 at 5:00

answered Dec 20, 2008 at 4:54

tvanfosson

532k ● 102 ● 699 ● 798

2

While you are absolutely correct that namespaces are good, wholesome and completely necessary, it sounds like considerable effort has been put forth to establish the status quo on your team. The only way I think you have a chance of selling the idea is to:

1. Set hard numbers on the cost of not having namespaces (maintainability, debugging, etc.) and get executive management on board.
2. Volunteer to create the namespacing conventions and do ALL the work to convert the projects over.

Unfortunately, in my experience once something as pervasive as namespace convention has been established, it takes an act of Congress to get it changed.

Share Improve this answer Follow

answered Dec 20, 2008 at 5:15

Dave Swersky

34.8k ● 9 ● 82 ● 120

1. Code legibility and organization, like many have pointed out.
2. Separation of similarly-named classes into different "spaces", like Network.Session, Meeting.Session, WebServer.Session, etc. In real world, there are many objects that bear the same or similar names. That's why names need their own spaces to qualify them. Namespaces also facilitate the naming of assemblies when used properly, like **System.Web.DLL** and **System.Messaging.DLL**. This helps code re-use, because you separate code into smaller functional chunks (DLLs) that can be included as needed, without including the whole giant mass (or mess).
3. Namespaces are boundaries that separate code of different projects, purposes and categories. In larger projects where many DLL files are shared and re-used, programmers need a easily-memorable way to reference the right classes, and to avoid referencing the wrong classes. Can one imagine the unmanageable madness if Microsoft had decided to put all the .NET classes into one single "System" namespace? How would any programmer even start to write code to invoke these classes? Furthermore, you would need to deal with classes named like **WindowsFormsTextBox** and **WebUIWebControlsTextBox** or something longer than that. Having proper namespaces helps programmers write code quickly and safely.

The benefit of namespaces become more obvious in larger projects, or when small projects gradually become large enough to be a issue. You can take a look at the size of the current projects, and determine how hard you need to twist your colleague's arms.

Share Improve this answer Follow

edited Dec 20, 2008 at 6:30

answered Dec 20, 2008 at 6:23

YYL

1

For some people, no amount of logical discourse will be quite as convincing as a physical analogy.

So, while everyone is at lunch one day, take all of their books out of their offices and bookcases, and pile them on the conference room table. Be sure to thoroughly mix up the computer books with the manga comics and the month-old MSDN magazines, etc.

Tell them that this is a physical representation of having one namespace - the conference table is the namespace, the books are the modules/classes/projects.

Then ask them to find the book on genetic algorithms in the pile. (make sure you hid it at the bottom of the pile).

If they still don't get it, give up and start looking for another job. You don't want to work with people that thick!

;-)

Share Improve this answer

Follow

answered Dec 20, 2008 at 5:04

Steven A. Lowe

61.1k ● 19 ● 135 ● 204

+1 Great analogy - but wouldn't it be deeper than finding the book on Genetic Algorithms? Wouldn't it be more like tearing the covers off all the books and telling them to try and find a chapter specifically on how to make use genetic algorithms in your application? – BenAlabaster Dec 20, 2008 at 17:59

why stop there? tear out all the book chapters and shuffle them up on the table ;-) – Steven A. Lowe Dec 20, 2008 at 19:04

0

If your code was laundry you would have two choices, either you can keep everything in one big pile and rifle through it every time you need something or you can put your clothes away in a bureau.

Share Improve this answer

Follow

answered Dec 20, 2008 at 4:19

Andrew Hare

351k ● 75 ● 645 ● 641

0

If I have a well designed namespace scheme, the using/Imports statements at the top of the code file documents what I'm using in this file.

answered Dec 20, 2008 at 4:21

Share Improve this answer

Follow

jrcs3

2,780 ● 1 ● 19 ● 35

If you use using/imports directives to import the contents of numerous namespaces into the current working namespace, does that negate the benefits of using namespaces in the first place? Or are there only certain namespaces that you're willing to import (e.g. core built-in language ones)? – reuben Dec 20, 2008 at 5:00

@Reuben: You only import the relevant namespaces for your project so if you've got multiple namespaces - the most obvious are the WebControls, HtmlControls and Forms namespaces all of which contain a class called Table...you would only import the one relevant for this portion of your project. – BenAlabaster Dec 20, 2008 at 17:55

0

If you have to convince them that having partitioned namespaces and not simply "The One True Namespace" is a good, desirable thing then that seems to indicate you probably will not be able to convince them of it. Just to be honest.

Share Improve this answer

Follow

answered Dec 20, 2008 at 5:04

BobbyShaftoe

28.5k ● 7 ● 54 ● 74

0

If the developers are comfortable with VB6 and with the structure of the existing VB6 project, it might make sense to have the ported project be structured a lot like the original, even if it compromizes some of the best

practices in .net. (I assume VB6 doesnt support namespaces?) Namespaces could be introduced later when the new code base is stable and the devlopers more comfortabe with .net.

Share Improve this answer

Follow

answered Dec 20, 2008 at 11:29

JacquesB

42.6k ● 13 ● 75 ● 88

0

What do they think namespaces are for? Why do they think the BCL uses them?

Share Improve this answer

Follow

answered Dec 20, 2008 at 19:35

Greg Dean

30k ● 15 ● 69 ● 79