

Re: C# Plugin system – same interface in two different assemblies...

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-01/msg02344.html>

- *From:* Jon Skeet [C# MVP] <skeet@xxxxxxxx>
 - *Date:* Thu, 17 Jan 2008 23:48:18 -0000
-

WTH <wth@xxxxxxxx> wrote:

All of those situations mean you've got the code in two places. That doesn't sound like good practice to me.

? Do you mean it's defined twice?

Yes – the source code for the interface is present twice.

But that's the whole point.

That's duplication, which it's good practice to avoid.

Define it

where and when you need it, don't force people to have it defined all the time like you would if they have to include an assembly to get one interface and that assembly contains 30 other interface definitions. The code is still there. If you simply put one interface per assembly you're only including in the developers code once but then they have to have all the assemblies which they 'may' use...

No, they only have to reference the assemblies they **do** use.

No need. Just put each plugin in its host's assembly, and reference that. If you have many plugins, you have complexity whatever you do. Duplicating source code just makes that worse, rather than better.

What on earth are you talking about? Put each plugin in its hosts assembly?
Did you mean interface?

Re: C# Plugin system – same interface in two different assemblies...

Yes, I meant the interface.

If you meant interface, that's a very strange way to deploy and SDK. What if you're application/host requires licensing? What if it has a complicated configuration? Serious deployment dependencies?

Who said anything about configuring it or installing it? You don't need to do either of those things just to *reference* it.

Of course, if it becomes a problem for any reason, you can separate out the interface into its own assembly – no problem.

<snip>

But code duplication is A-OK by you is it? Sorry, I really don't buy it.

You mean multiple definitions of the interface? Do you not realize that this happens no matter what method you choose? Are you worried about typos or something? lol.

When somebody re-uses a class that someone else wrote, do you consider that code duplication and you don't "buy it"?

No, I don't consider that code duplication. The source code only lives in one place. Changes to it only have to occur in one place, then get deployed everywhere it's used. I consider that better than having the source code everywhere that the interface would be used. I'm rather glad I don't have to include the interface code for IDisposable, IEnumerable<T>, IList<T> etc in every project which uses them.

The latter is *much* better IMO.

Well, in my experience in supporting the exact scenarios I have described, it is far easier to develop, support, and extend via the former.

Your experience which is limited in C#, by your own admission.

I'm afraid I see having to duplicate source code as a much bigger wart than compiling against a fixed binary.

Re: C# Plugin system – same interface in two different assemblies...

Re: C# Plugin system – same interface in two different assemblies...

You seem to have latched onto this as your only reasoning as to why C# cannot discern that two interfaces are identical.

Nope – it's just *one* good reason. It's a good enough one for me. Various other reasons are also available, but I only need the one.

I'd hate to think of how you share code with others, presumably you just send them an assembly and if there's a problem with it, they have to ask you to fix it and ship them a new assembly.

I suspect that most people who use my open source library only want or need the DLL. Those who want to change the source can do so – but I'd expect they'd usually still build to a DLL and reference that. Assemblies are the units of code reference in .NET – not source code.

Hopefully you're sitting around with nothing to do when this happens. Here we can't do that. If there's a bug in an SDK client helper class, the sdk developer can modify it because he has the file, not the assembly. He doesn't have to wait for a new version of the SDK to come out in 6 months to a year.

No, but he then has to patch it every time there's a new release of the SDK. Meanwhile, he may have introduced other bugs. The whole thing's a versioning nightmare.

I don't expect to convince you, to be honest.

--

Jon Skeet – <skeet@xxxxxxxx>

<http://www.pobox.com/~skeet> Blog: <http://www.ms MVPs.com/jon.skeet>

World class .NET training in the UK: <http://iterativetraining.co.uk>

.