

Re: net 3 replaces net 1?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.general/2006-12/msg00819.html>

- *From:* "Cor Ligthert [MVP]" <notmyfirstname@xxxxxxxxxx>
 - *Date:* Sun, 24 Dec 2006 09:39:11 +0100
-

Nick,

Were did you check this, at office. I hope (and know) that this is not an official message from Microsoft.

I am really very much in doubt if it is true, Microsoft has promised with the start of the frameworks that they would not create again a new kind and even worse DLL hell., It would be contantly downwards compatible, we the exception of the development part. What is logic, because than it would be in fact upwards compatibility..

Cor

"Nick Malik [Microsoft]" <nickmalik@xxxxxxxxxxxxxxxxxxxx> schreef in bericht news:YPidnbL5G4nvHRDYnZ2dnUVZ_uOmnZ2d@xxxxxxxxxxxxxxxx

Hi Cor,

"Cor Ligthert [MVP]" <notmyfirstname@xxxxxxxxxx> wrote in message [news:Om\\$iek\\$IHHA.816@xxxxxxxxxxxxxxxxxxxxxxxx](news:OmiekIHHA.816@xxxxxxxxxxxxxxxxxxxxxxxx)

Peter,

Are you sure of that?

Peter is correct. If an app is compiled for 1.1, and you are a 'user' of that app, then you have to keep 1.1 installed on your machine. If you are the 'author/programmer' of that app, on the other hand, you can install a newer version of visual studio, recompile, and then consume .net 2.0 or .net 3.0 (which is the addition of a slew of classes to .net 2.0). In that case, the new DLL you create would not need .net 1.1.

In my idea would 2.0 replace completely version 1.x the problem is that there are some bugs in that area or better no backward compatibility of

Re: net 3 replaces net 1?

old bugs was concerned in.

I think you are asking why some issues were not 'addressed' when 2.0 was released. The reason being that those issues are nearly all involved in "tradeoffs in behavior."

In other words, a class could behave in mechanism (1) or mechanism (2). Mechanism (1) was chosen and has side effect (x). People who don't like side effect (x) would prefer that MS had chosen mechanism (2). However, other developers who don't experience side effect (x) or don't care about it have written code that depends on Mechanism (1). Therefore, switching to Mechanism (2) in the 2.0 version of the framework would break backwards compatibility.

There are always tradeoffs. That is life. I think the basic premise goes something like this: You are always free to write a new class that implements Mechanism (2) and make it available as part of a third-party library. MS had to pick one. Picking two is bad.

Maybe is that the reason that I get now on bug reports to much messages in my eyes that say that they cannot be resolved because of backward compatibility.

Cor

I hope that helps,

--

--- Nick Malik [Microsoft]
MCSD, CFPS, Certified Scrummaster
<http://blogs.msdn.com/nickmalik>

Disclaimer: Opinions expressed in this forum are my own, and not representative of my employer.

I do not answer questions on behalf of my employer. I'm just a programmer helping programmers.

--