

Re: Garbage collection

Source: <http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.faqs/2006-09/msg00031.html>

- *From:* Jon Shemitz <jon@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Tue, 26 Sep 2006 19:10:37 -0700
-

"Scott M." wrote:

Yes, but calling Dispose on every class that exposes it will cause additional CPU costs, not less.

No, in most cases it's spending time to save time. When an IDisposable class has a finalizer, its Dispose methods typically runs the exact same code that the finalizer runs, plus it calls GC.SuppressFinalize. This is saving all the CPU costs involved in running the finalizer, plus it allows the memory to be reused sooner.

When an IDisposable class does not have a finalizer, it usually has private reference(s) to handle class(es) that do have a finalizer, so the call to Dispose allows it to close its handle(s), thus (again) saving all the CPU costs involved in running the handle finalizer(s), plus allowing the memory to be reused sooner.

The only time calling Dispose doesn't save time is with the minority of IDisposable implementations that aren't avoiding a finalizer, that are doing something like restoring the GUI cursor or noting how long it's been since a stopwatch class has been created.

So, by calling Dispose on every object that implements it, you are reducing memory consumption and garbage collection costs.

You mean you are increasing memory consumption and garbage collection costs, not reducing them.

No, I meant what I said. Invoking IDisposable allows data to be closed when you're done with, and thus avoids the use of the finalizer. The

Re: Garbage collection

finalizer is a fail-safe, but using it adds (avoidable) relocation costs and means that the gc has to look at the dead object (at least) twice.

—

..NET 2.0 for Delphi Programmers www.midnightbeach.com/.net
Delphi skills make .NET easy to learn Great reviews & good sales.

.