

## Re: Disposing managed resources

---

*Source:*

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2007-10/msg00405.html>

---

- *From:* Varangian <[ofmars@xxxxxxxx](mailto:ofmars@xxxxxxxx)>
  - *Date:* Wed, 03 Oct 2007 11:24:35 -0000
- 

On Oct 3, 12:37 pm, "Laura T." <[LT\\_s...@xxxxxxxx](mailto:LT_s...@xxxxxxxx)> wrote:

"Varangian" <[ofm...@xxxxxxxx](mailto:ofm...@xxxxxxxx)> ha scritto nel  
[messaggionews:1191402348.513390.264650@xx](mailto:messaggionews:1191402348.513390.264650@xx)

On Oct 3, 9:49 am, "Laura T." <[LT\\_s...@xxxxxxxx](mailto:LT_s...@xxxxxxxx)> wrote:

In fact, if the resource is "managed", then heck, let it be managed. :-)

The only case where disposing managed resources *\*may\** be useful is "acquire late"/"release early" scenario. That is, from the performance point of view, if you know that a big (big) memory structure is not needed anymore but normally it would still be rooted, it may help GC to dispose it and release references early so that GC can, if it needs, to get more resources. But it's not deterministic. See <http://blogs.msdn.com/brada/archive/2004/05/24/140645.aspxandlinks>.

"Peter Duniho" <[NpOeStPe...@xxxxxxxxxxxxxxxxxxxxxxxx](mailto:NpOeStPe...@xxxxxxxxxxxxxxxxxxxxxxxx)> ha scritto nel  
[messaggionews:13g5o4hhmq4gi46@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:messaggionews:13g5o4hhmq4gi46@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

Varangian wrote:

## Re: Disposing managed resources

was wondering of how to  
dispose of managed  
resources?

It depends. Some classes implement  
IDisposable, and in some of those  
cases, there are managed resources that can  
be "released"  
(unreferenced)  
using the Dispose() method of that interface.

But I think that normally, there's not any  
need to dispose of managed  
resources. It's the unmanaged ones you care  
about. Managed ones are,  
by  
virtue of being managed, going to be cleaned  
up when there's a good  
reason  
to clean them up, and otherwise left alone.

Since you shouldn't waste time cleaning  
something up when there's not a  
good reason to, I think that for managed  
resources, just managing the  
lifetime of the resource relative to your own  
use is sufficient.

More typically, you would use IDisposable  
to dispose of a resource that  
is  
either unmanaged, or itself has unmanaged  
resources and thus implements  
IDisposable.

or referencing every  
member of a class to null  
will release  
resources...?

## Re: Disposing managed resources

Setting references to null can allow a resource to be released. But it doesn't necessarily do so right away. And I think that for managed resources, you shouldn't even care whether they are or not.

<http://www.marcclyton.com/tabid/79/Default.aspx>

This seems to discuss the use of IDisposable to release unmanaged resources. It seems to me, however, that the author doesn't really understand the .NET memory model very well, nor does he really seem to understand the true purpose of IDisposable (and he also doesn't seem to understand that when you read a compressed image file to memory, of course it has to be decompressed to be used and so occupies much more memory than it does on disk...but that's a whole other issue :)).

[http://www.codeproject.com/managedcpp/garbage\\_collection.asp-sources](http://www.codeproject.com/managedcpp/garbage_collection.asp-sources)

This article seems much better, but I'm not clear on how it is specifically related to your question. It doesn't seem to me to discuss the question of disposing managed resources per se, though it does discuss IDisposable.

Pete

## Re: Disposing managed resources

there are so many sources about how to use the IDisposable.... that's what i thought that an image is an unmanaged resource.

The managed System.Drawing.Image holds an unmanaged resource (GDI handle), so it must be disposed.

It's not 100% managed. Almost all System.Drawing hold unmanaged resources and thus must be disposed as soon as possible.

Same goes for System.IO.\* and many others.

so how the GC.SuppressFinalize comes into use? I read that this may improve slightly insignificantly the performance... so that managed object doesn't need to be finalized again because of its managed resource. how would the SuppressFinalize come into use?

GC.SuppressFinalize() is intended only for Dispose(). It tells CLR that that "I'm done. No more cleanup needed, thanks", by clearing a virtual "sorry, could you please clean me up because someone forgot to do it?" flag. CLR says "OK, thanks for the cooperation. Now I can just deallocate you which is faster than remembering to call your Finalizer(), call it and then deallocate. Good."

If you don't call (but you do) Dispose(), the object still has the "sorry, could you please clean me up because someone forgot to do it?" flag set, so CLR will take the trouble (but it makes it angry) to call the Finalizer() (someday) to clear the flag and then deallocate.

So from your point of view (consumer of the object), you do not need (I'd say must not) to call GC.SuppressFinalize() (taken that the Dispose() was written correctly).

It won't help.

thanks Laura .T.

the last part I didn't get it quite correctly... you said (taken that the Dispose() was written correctly) how can you say that if the Dispose method will supposedly do the SuppressFinalize?

.