

Re: GC and dispose

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.framework/2006-06/msg00636.html>

- *From:* "Carl Daniel [VC++ MVP]" <cpdaniel_remove_this_and_nospam@xxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 21 Jun 2006 16:12:20 -0700
-

"Göran Andersson" <guffa@xxxxxxxxxx> wrote in message
news:evglUFYIGHA.2128@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Markus Stoeger wrote:

Simon Hart wrote:

Agree. We often use the Dispose (IDisposable) for cleaning managed resources as well as unmanaged. We tend to use the Finalizer (Destructor) for unmanaged deallocation.

Could you explain why you prefer the finalizer instead of a manual call to Dispose for freeing unmanaged resources?

As far as I know you can never tell when the GC will finalize an object. So it might keep floating around forever as well. Which means that you could run out of handles for example.

Maybe I've misunderstood something behind GC and the finalizer?.

Max

No, that is correct.

The Dispose method should be used to free resources. The finalizer should call Dispose as a backup if the program failed to call it.

More specifically, the recommended structure for Finalize and Dispose is:

```
class C : IDisposable
{
void IDisposable.Dispose()
{
Dispose(true);
```

Re: GC and dispose

```
}  
  
~C()  
{  
Dispose(false)  
}  
  
protected virtual void Dispose(bool disposing)  
{  
if (disposing)  
{  
// free managed resources here  
  
GC.SuppressFinalize(this);  
}  
  
// free unmanaged resources here  
}  
}
```

The C++/CLI compiler automatically implements this structure, while in C#/VB you have to do it yourself. Current versions of FxCop will raise warnings if your class implements IDisposable but doesn't do it this way.

Of course, if your class cannot be derived from, then there's no need for Dispose(bool) to be virtual.

-cd

-cd

.