

Re: Simple Dispose Question

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2005-02/6091.html>

From: Jeffrey Palermo, MCAD.Net (jeffreypalermo_at_gmail.com)

Date: 02/24/05

Date: 24 Feb 2005 06:16:19 -0800

Jeff,

I'd like to add something that hasn't come out in this thread yet. The use of `.Dispose()` has been thoroughly discussed, but I'd like to answer your question regarding the benefit of additionally setting the variable to null.

If I assume that you have a method containing only that code:

```
void Foo(){
  DataAdapter da = null;
  try {
    // Some logic here...
  } catch (Exception ex) {
    // Some exception handling logic here...
  } finally {
    // Clean up...
    if (da != null) {
      da.Dispose();
      da = null;
    }
  }
}
```

you are setting `da` to null one statement before it would naturally be set to null (or go out of scope). At the end of every method, every variable declared in the method is set to null and goes out of scope, so there is no benefit of explicitly setting it to null because it will automatically happen one statement later.

On the other hand, there are situations where it may be beneficial to set this to null. For instance:

```
void Foo(){
  DataAdapter da = new DataAdapter(. . .);
  try{
    // use it somehow
  }finally{
    da.Dispose();
  }
  // now call another method that may take a while to return
}
```

Here, we would wait for another long-running method to return and the da variable would still be in scope. Depending on what the long-running method does, the GC may run several times. IN THIS CASE, if you set da to null just before calling the long-running method, any garbage collections would be able to reclaim that memory, and you would have a benefit.

Normally, my methods are small enough that my variables naturally go out of scope quickly, but in cases where they might not, set it to null.

Best regards,
Jeffrey Palermo
Blog: <http://www.jeffreypalermo.com>