

Re: What's finally keyword good for?

Source:

<http://www.tech-archive.net/Archive/DotNet/microsoft.public.dotnet.languages.csharp/2008-12/msg00889.html>

- *From:* "Michael D. Ober" <obermd.@alum.mit.edu.nospam>
 - *Date:* Mon, 8 Dec 2008 18:53:41 -0700
-

"Ollis" <Ollis@xxxxxxxx> wrote in message <news:uJ3pUqZWJHA.4820@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>

Christophe Lephay wrote:

"Ollis" <No@xxxxxx> a écrit dans le message de groupe de discussion :
E79754C2-9B05-4210-AFA6-5EF4DB0FC9B2@xxxxxxxxxxxxxxxxxxxx

I never said anything about catch all exceptions. That came from you not me.

Well, that came from the code you posted, but i don't doubt you don't write this kind of code in production.

I write code for BLL and DAL, and I don't write the kind of code you are talking about that needs to be looking at a bunch of exceptions. One shoe doesn't fit all situations.

Why start looking at a bunch of exceptions when one is looking for a particular exception, knowing that the code is only going to throw a certain type if exception?

I didn't know I had to elaborate as to why, based on the code I posted. I thought it was pretty straight forward. I don't like using the **finally**, and one doesn't have to use the **finally** to clean things up. It's kind of simple. I like simple code. I have no need for complicated code.

If you want to use the **finally**, then use it. It doesn't mean it's a best practice.

And all that was stated by me is what was that not knowing about all exceptions in a given situation was **poor** programming.

I'm not so sure. Many exceptions are for debugging purpose (invariants checking) and, in this case, the user has no responsibility about them.

Re: What's finally keyword good for?

As far as I am concerned, an exception serves one purpose, which is to control the path of code execution to continue processing or terminate processing.

I have no other use for exceptions.

I guess you'll make
of it what you want in the defense of this other person.

???

I don't have much use of finally either,
because most of resources are
managed and i prefer using(...) otherwise.
However, finally and using are
equivalent (as long as the class is
IDisposable), but the code you posted
was not.

You made much to do about the code I have posted. Like I said, I don't like using the
finally, the code I posted by no means was the gospel. It was just an example --- no more
or no less.

This is one thing I don't like about this NG as people here tend to take a simple example of
something, blow it way out of proportion and blow it to the Moon.

That's not true. You use a USING statement on an
ADO.NET connection to a
database server like Oracle and it blows with the connection
open. The
application is keeping state while continuously coming back to
open a new
connection in the process. All that's going to happen is the
database server
is going to run out of connections, unless one does
something to close the
connection during the abort each time, because that USING
is not doing it.

Hum. It seems to me very unlikely that the behavior you describe could be
related to the using instruction.

You mean you have never faced that bullet. A USING statement to make a database

Re: What's finally keyword good for?

connection and nothing aborts is OK with *close* and *dispose* of the connection. One has a SQL command abort inside the USING with a open connection and no *close* and *dispose* occurs, and the connection is left open. It is what I have experienced, and the bullet I have faced.

Finally is extremely useful even if there are no errors. If your routine exits early and needs to perform any cleanup, use the code template below. Using finally in this way allows you to exit your routine anywhere it makes logical sense while allowing you to put your clean up code in a single location in the function. For those of you who say – this is exactly what the Using statement block does, you're correct. This is simply a more general example that doesn't depend on the developer of your resources implementing Dispose.

```
int foo()
{
int result;
try {
// Open resources required for the processing
// do some processing
if (earlytermination) {
return result;
}
} finally {
// perform cleanup of open resources
}
}
```

Mike.

.