Should I catch exceptions only to log them?

Asked 16 years, 3 months ago Modified 2 years ago Viewed 18k times

31

Should I catch exceptions for logging purposes?

```
public foo(..)
{
   try
   {
     ...
   } catch (Exception ex) {
     Logger.Error(ex);
     throw;
   }
}
```

If I have this in place in each of my layers (DataAccess, Business and WebService) it means the exception is logged several times.

Does it make sense to do so if my layers are in separate projects and only the public interfaces have try/catch in them? Why? Why not? Is there a different approach I could use?

c# logging exception

Share

asked Sep 18, 2008 at 18:44

Improve this question

Follow

Xerx

3,775 ● 8 ● 34 ● 30

16 Answers

Sorted by: Highest score (default)

39

Definitely not. You should find the correct place to **handle** the exception (actually do something, like catch-and-not-rethrow), and then log it. You can and should include the entire stack trace of course, but following your suggestion would litter the code with try-catch blocks.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:47

ripper234

230k ● 280 ● 642 ● 913

20

Unless you are going to change the exception, you should only log at the level where you are going to handle the error and not rethrow it. Otherwise your log just has a bunch of "noise", 3 or more of the same message logged, once at each layer.

My best practice is:

1. Only try/catch in public methods (in general; obviously if you are trapping for a specific error you would check for it there)
2. Only log in the UI layer right before suppressing the error and redirecting to an error page/form.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:48

Guy Starbuck
21.8k ● 7 ● 54 ● 64

11

The general rule of thumb is that you only catch an exception if you can actually do something about it. So at the Business or Data layer, you would only catch the exception in situation's like this:

```
try
{
    this.Persist(trans);
}
catch(Exception ex)
{
    trans.Rollback();
    throw;
}
```

My Business/Data Layer attempts to save the data - if an exception is generated, any transactions are rolled back and the exception is sent to the UI layer.

At the UI layer, you can implement a common exception handler:

```
Application.ThreadException += new
ThreadExceptionEventHandler(Application_ThreadExceptio
```

Which then handles all exceptions. It might log the exception and then display a user friendly response:

```
static void Application_ThreadException(object sender,
e)
{
    LogException(e.Exception);
}

static void LogException(Exception ex)
{
    YYYExceptionHandling.HandleException(ex,
        YYYExceptionHandling.ExceptionPolicyType.YYY_P
        YYYExceptionHandling.ExceptionPriority.Medium,
        "An error has occurred, please contact Adminis
}
```

In the actual UI code, you can catch individual exception's if you are going to do something different - such as display a different friendly message or modify the screen, etc.

Also, just as a reminder, always try to handle errors - for example divide by 0 - rather than throw an exception.

Share Improve this answer

Follow

edited Dec 20, 2022 at 6:52

zwcloud

4,879 ● 3 ● 44 ● 76

answered Sep 18, 2008 at 19:04

Pat

299 ● 2 ● 7

17 As an aside, do not do `throw ex` to rethrow. Just do `throw` otherwise you will lose the stacktrace from the origiinal exception. – 1800 INFORMATION Sep 18, 2008 at 19:42

2

It's good practice is to **translate the exceptions**. Don't just log them. If you want to know the specific reason an exception was thrown, throw specific exceptions:

```
public void connect() throws ConnectionException {
   try {
       File conf = new File("blabla");
       ...
   } catch (FileNotFoundException ex) {
       LOGGER.error("log message", ex);
       throw new ConnectionException("The configuratio
ex);
   }
}
```

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:56

Marcio Aguiar

14.5k ● 6 ● 40 ● 42

He, wasn't this a c# question? – ɔɔɯəɹ Apr 28, 2013 at 23:24

2

Sometimes you need to log data which is not available where the exception is handled. In that case, it is appropriate to log just to get that information out.

For example (Java pseudocode):

```
public void methodWithDynamicallyGeneratedSQL() throws
    String sql = ...; // Generate some SQL
    try {
        ... // Try running the query
    }
```

```
    catch (SQLException ex) {
        // Don't bother to log the stack trace, that w
        // be printed when the exception is handled fo
        logger.error(ex.toString()+"For SQL: '"+sql+"'
        throw;  // Handle the exception long after the
    }
}
```

This is similar to retroactive logging (my terminology), where you buffer a log of events but don't write them unless there's a trigger event, such as an exception being thrown.

Share Improve this answer

Follow

edited Dec 20, 2022 at 6:52

zwcloud

4,879 ● 3 ● 44 ● 76

answered Oct 23, 2008 at 22:06

David Leppik

3,304 ● 1 ● 32 ● 18

1

Use your own exceptions to wrap inbuild exception. This way you can distinct between known and unknown errors when catching exception. This is usefull if you have a method that calls other methods that are likely throwing excpetions to react upon expected and unexpected failures

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:49

GHad

9,621 ● 6 ● 36 ● 40

1

you may want to lookup standard exception handling styles, but my understanding is this: handle exceptions at the level where you can add extra detail to the exception, or at the level where you will present the exception to the user.

in your example you are doing nothing but catching the exception, logging it, and throwing it again.. why not just catch it at the highest level with one try/catch instead of inside every method if all you are doing is logging it?

i would only handle it at that tier if you were going to add some useful information to the exception before throwing it again - wrap the exception in a new exception you create that has useful information beyond the low level exception text which usually means little to anyone without some context..

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:52

SmartyP

1,359 ● 1 ● 8 ● 14

0

If you're required to log all exceptions, then it's a fantastic idea. That said, logging all exceptions without another reason isn't such a good idea.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:49

hometoast

11.8k • 5 • 42 • 58

0

You may want to log at the highest level, which is usually your UI or web service code. Logging multiple times is sort of a waste. Also, you want to know the whole story when you are looking at the log.

In one of our applications, all of our pages are derived from a BasePage object, and this object handles the exception handling and error logging.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:49

Charles Graham

24.8k • 14 • 47 • 56

0

If that's the only thing it does, i think is better to remove the try/catch's from those classes and let the exception be raised to the class that is responsible on handling them. That way you get only one log per exception giving you more clear logs and even you can log the stacktrace so you wont miss from where the exception was originated.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:50

Lucas S.

13.5k ● 9 ● 48 ● 46

I am not the thread maker, but how do you let the exception be raised? Just by writing throw or anything else?
– GurdeepS Mar 22, 2010 at 22:44

0

My method is to log the exceptions only in the handler. The 'real' handler so to speak. Otherwise the log will be very hard to read and the code less structured.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:53

Karl

3,216 ● 1 ● 23 ● 29

0

It depends on the Exception: if this actually should not happen, I definitely would log it. On the other way: if you expect this Exception you should think about the design of the application.

Either way: you should at least try to specify the Exception you want to rethrow, catch or log.

```
public foo(..)
{
   try
   {
      ...
```

```
    }
    catch (NullReferenceException ex) {
      DoSmth(e);
    }
    catch (ArgumentExcetion ex) {
      DoSmth(e);
    }
    catch (Exception ex) {
      DoSmth(e);
    }
}
```

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:54

MADMap

3,192 ● 4 ● 26 ● 32

0

You will want to log at a tier boundary. For example, if your business tier can be deployed on a physically separate machine in an n-tier application, then it makes sense to log and throw the error in this way.

In this way you have a log of exceptions on the server and don't need to go poking around client machines to find out what happened.

I use this pattern in business tiers of applications that use Remoting or ASMX web services. With WCF you can intercept and log an exception using an IErrorHandler attached to your ChannelDispatcher (another subject entirely) - so you don't need the try/catch/throw pattern.

Share Improve this answer

Follow

answered Sep 18, 2008 at 18:59

to StackOverflow

125k ● 33 ● 210 ● 341

0

You need to develop a strategy for handling exceptions. I don't recommend the catch and rethrow. In addition to the superfluous log entries it makes the code harder to read. Consider writing to the log in the constructor for the exception. This reserves the try/catch for exceptions that you want to recover from; making the code easier to read. To deal with unexpected or unrecoverable exceptions, you may want a try/catch near the outermost layer of the program to log diagnostic information.

BTW, if this is C++ your catch block is creating a copy of the exception object which can be a potential source of additional problems. Try catching a reference to the exception type:

```
catch (const Exception& ex) { ... }
```

Share Improve this answer

Follow

answered Sep 18, 2008 at 19:48

Diastrophism

15.4k ● 1 ● 18 ● 7

0

[This](#) Software Engineering Radio podcast is a very good reference for best practices in error handling. There are actually 2 lectures.

Share Improve this answer

Follow

answered Sep 18, 2008 at 20:37

Andrew Cowenhoven

2,798 ● 23 ● 28

It's bad practice in general, unless you need to log for very specific reasons.

With respect in general log exception, it should be handled in root exception handler.

0

Share Improve this answer Follow

edited Feb 4, 2021 at 16:28

answered Feb 4, 2021 at 16:10

Rezwan4029

109 • 3