

Forum:[Hack Forums](#)

Thread: [\[C#\]\[SOURCE\] DeadLine's File Binder](#)

DeadLine - April 15, 2012, 10:50 am

Hello,

DeadLine's File Binder is an easy to use file binder that you can use to bind 2 files into one executable file.

Screenshot:

[img]<http://i363.photobucket.com/albums/oo73/Allibobo/fb1.png>[/img]

Screenshot:

[img]<http://i363.photobucket.com/albums/oo73/Allibobo/fb2.png>[/img]

Download:

<http://www.mediafire.com/?ghkhzk55fipo4ih>

VirusTotal:

<https://www.virustotal.com/file/88898ead45b1bf225f525874914d5827fbb6237be053ca1a5c08f66b1ecf8601/analysis/1334512165/>

Please leave a comment to keep this thread alive !

DeadLine

skyline7349 - April 15, 2012, 11:09 am

thanks man taking look at it now. i really want to know how to theme a program.

triplex-x - April 15, 2012, 9:40 pm

nice share bro and thanks for sharing the sources. awesome binder and the GUI

Vuln - April 29, 2012, 4:18 am

Why do you put like literally everything in a try/catch statement? o.0

PHP Code

```
<?php
try
{
    if (chbMessageBox.Checked == true)
    {
        grbMessageBox.Enabled = true;
    }
    else
    {
        grbMessageBox.Enabled = false;
```

```
}  
}  
catch  
{  
}
```

^ for example, what error could possibly be caught?

chbMessagebox.Checked == true could be shortened to chbMessagebox.Checked btw.

Also keep in mind that the output file still has the assembly info of the stub, which probably causes a detection or two :p

Anyways, it's working fine, good job. Nice website of yours too.

DeadLine - April 29, 2012, 5:33 am

(Apr 29, 2012 04:18 AM)makulu Wrote: [\[CITE\]](#) Why do you put like literally everything in a try/catch statement? o.0

PHP Code

```
<?php  
try  
{  
    if (chbMessagebox.Checked == true)  
    {  
        grbMessagebox.Enabled = true;  
    }  
    else  
    {  
        grbMessagebox.Enabled = false;  
    }  
}  
catch  
{  
}
```

^ for example, what error could possibly be caught?

chbMessagebox.Checked == true could be shortened to chbMessagebox.Checked btw.

Also keep in mind that the output file still has the assembly info of the stub, which probably causes a detection or two :p

Anyways, it's working fine, good job. Nice website of yours too.

Hello,

I use try/catch because you never know when an error may appear. The codes might be good, but there can always be a problem when an advanced user starts editing the functions with an editing program and causes an error. Also, it is good practice :)

And thank you !

Kind regards,

DeadLine

+Bourbon - April 29, 2012, 6:34 am

(Apr 29, 2012 05:33 AM)DeadLine Wrote: [\[CITE\]](#) Hello,

I use try/catch because you never know when an error may appear. The codes might be good, but there can always be a problem when an advanced user starts editing the functions with an editing program and causes an error. Also, it is good practice :)

And thank you !

Kind regards,

DeadLine

You're using them wrong. You shouldn't use try catch blocks to mask errors. You should trap errors and perform statements accordingly. What's strange is that you have proper error trapping and then surround the statement in a try catch clause regardless. It's not a good programming practice it's very poor. There's a slight performance draw back with testing for an exception but the real performance draw back would be using try catch blocks instead of error trapping. Not really relevant to you though.

Sometimes using a try catch block is essential as there are no means for error trapping. For example, when you're using a Stream Reader:

```
Code .   StreamReader Reader;  
        Reader = new StreamReader("InvalidPath");  
        Reader.ReadToEnd();
```

As far as i know there's no way to determine if the file isn't being used by another process or whatever condition that may cause an exception. So most appropriate thing to do would be handle an IOException thrown by the StreamReader class constructor.

An example of where a try catch block is redundant can be found in seconds using your project.

DeadLine - April 29, 2012, 6:40 am

(Apr 29, 2012 06:34 AM)ByteBlast Wrote: [\[CITE\]](#) You're using them wrong. You shouldn't use try catch blocks to mask errors. You should trap errors and perform statements accordingly. What's strange is that you have proper error trapping and then surround the statement in a try catch clause regardless. It's not a good programming practice it's very poor. There's a slight performance draw back with testing for an exception but the real performance draw back would be using try catch blocks instead of error trapping. Not really relevant to you though.

Sometimes using a try catch block is essential as there are no means for error trapping. For example, when you're using a Stream Reader:

```
Code .   StreamReader Reader;  
        Reader = new StreamReader("InvalidPath");  
        Reader.ReadToEnd();
```

As far as i know there's no way to determine if the file isn't being used by another process or whatever condition that may cause an exception. So most appropriate thing to do would be handle an IOException thrown by the StreamReader class constructor.

An example of where a try catch block is redundant can be found in seconds using your project. I already knew this, but thanks anyway.

+Bourbon - April 29, 2012, 6:45 am

(Apr 29, 2012 06:40 AM)DeadLine Wrote: [\[CITE\]](#) I already knew this, but thanks anyway. That explains why you use try catch blocks incorrectly.

dreamer66 - April 29, 2012, 7:46 am

There is nothing wrong with using try/catch blocks where it's unlikely to get any errors. Try/catch only affects performance when they fire so the performance loss is non existent if they don't.

DeadLine - April 29, 2012, 8:17 am

(Apr 29, 2012 07:46 AM)dreamer66 Wrote: [\[CITE\]](#) There is nothing wrong with using try/catch blocks where it's unlikely to get any errors. Try/catch only affects performance when they fire so the performance loss is non existent if they don't.

Finally someone who understands. Thank you sir !

+Bourbon - April 29, 2012, 9:00 am

(Apr 29, 2012 07:46 AM)dreamer66 Wrote: [\[CITE\]](#) There is nothing wrong with using try/catch blocks where it's unlikely to get any errors. Try/catch only affects performance when they fire so the performance loss is non existent if they don't.

(Apr 29, 2012 08:17 AM)DeadLine Wrote: [\[CITE\]](#) Finally someone who understands. Thank you sir ! I did a benchmark and it is in fact slower. But as mentioned before it's minuscule. Really bad conventions however.

DeadLine - April 29, 2012, 9:06 am

(Apr 29, 2012 09:00 AM)ByteBlast Wrote: [\[CITE\]](#) I did a benchmark and it is in fact slower. But as mentioned before it's minuscule. Really bad conventions however.

That is true, but once again, what if someone with a decompiler decided to edit your application and delete a checkbox and recompile it. It would cause an error when he binds files.

So in fact, I don't mind the application being 0,001 milliseconds slower than normal.

I know that this is a bad example but it is really all about being 100% sure that no error messages will pop up.

dreamer66 - April 29, 2012, 9:13 am

(Apr 29, 2012 09:00 AM)ByteBlast Wrote: [\[CITE\]](#) I did a benchmark and it is in fact slower. But as mentioned before it's minuscule. Really bad conventions however.

How did you do the benchmark exactly ?

```
Code Stopwatch sw = new Stopwatch();
    sw.Start();
```

```
    if (checkBox1.Checked == true)
        textBox1.Enabled = false;
```

```
    Console.WriteLine(sw.Elapsed);
    sw.Stop();
    textBox1.Enabled = true;
```

```
    try
    {
        if (checkBox1.Checked == true)
            textBox1.Enabled = false;
```

```
    Console.WriteLine(sw.Elapsed);
```

```
sw.Stop();  
}
```

```
catch { }
```

The results are almost identical

00:00:00.0001939

00:00:00.0002231

Needless to add, we are talking about values that are less than a millisecond.

+Bourbon - April 29, 2012, 9:26 am

(Apr 29, 2012 09:13 AM)dreamer66 Wrote: [\[CITE\]](#) How did you do the benchmark exactly ?

```
Code Stopwatch sw = new Stopwatch();  
sw.Start();
```

```
if (checkBox1.Checked == true)  
textBox1.Enabled = false;
```

```
Console.WriteLine(sw.Elapsed);  
sw.Stop();  
textBox1.Enabled = true;
```

```
try  
{  
if (checkBox1.Checked == true)  
textBox1.Enabled = false;
```

```
Console.WriteLine(sw.Elapsed);  
sw.Stop();  
}
```

```
catch { }
```

The results are almost identical

00:00:00.0001939

00:00:00.0002231

Needless to add, we are talking about values that are less than a millisecond.

Yes, minuscule.. Like I said. What you said was:

(Apr 29, 2012 07:46 AM)dreamer66 Wrote: [\[CITE\]](#) performance loss is non existent
So I was right.

(Apr 29, 2012 09:06 AM)DeadLine Wrote: [\[CITE\]](#) That is true, but once again, what if someone with a decompiler decided to edit your application and delete a checkbox and recompile it. It would cause an error when he binds files.

That's hilarious.

Vuln - April 29, 2012, 9:27 am

(Apr 29, 2012 09:06 AM)DeadLine Wrote: [\[CITE\]](#) That is true, but once again, what if someone with a decompiler decided to edit your application and delete a checkbox and recompile it. It would cause an error when he binds files.

So in fact, I don't mind the application being 0,001 milliseconds slower than normal.

I know that this is a bad example but it is really all about being 100% sure that no error messages will pop up.

Why in earth would someone decompile a binder to remove a checkbox? I know what you mean but that was probably the worst example ever... It's like... when a program saves a log every 0.5 seconds because a thunderstrike might disable the electricity one day :p

Anyways, no -visible- performance loss isn't a reason to do it, right? I mean, it takes longer to code, it's a bit harder to read, it's harder to see when a 'good' try/catch statement is used etc.

But enough about try/catch :p What did you do to bypass TR/Dropper.Gen? I can't get rid of it in my own binder yet the source is nearly the same.

dreamer66 - April 29, 2012, 9:30 am

If you are considering the difference 0.1 of a millisecond as a performance loss then clearly our way of thinking isn't the same. You can't even blink that fast.

To make it more graphically, you will need to use 3500 of those try/catch blocks to reach the eye blinking speed.. ;p

+Bourbon - April 29, 2012, 9:34 am

(Apr 29, 2012 09:30 AM)dreamer66 Wrote: [\[CITE\]](#) If you are considering the difference 0.1 of a millisecond as a performance loss then clearly our way of thinking isn't the same. You can't even blink that fast.

Quote:

minuscule: very small; "a minuscule kitchen"; "a minuscule amount of rain fell".

Not non existent however. And it's not a fixed length it's going to vary depending on the code. Could range from 1ms to x ms

dreamer66 - April 29, 2012, 9:41 am

(Apr 29, 2012 09:34 AM)ByteBlast Wrote: [\[CITE\]](#) Not non existent however. And it's not a fixed length it's going to vary depending on the code. Could range from 1ms to x ms
Here's something that might make you consider something :p

```
Code int count = 0;
```

```
    Stopwatch sw = new Stopwatch();  
    sw.Start();
```

```
    while (count != 500)  
        count++;
```

```
    Console.WriteLine(sw.Elapsed);  
    sw.Reset();
```

```
    count = 0;  
    sw.Start();  
    try  
    {
```

```
while (count != 500)
count++;
```

```
Console.WriteLine(sw.Elapsed);
sw.Stop();
}
```

```
catch { }
```

Also the first code didn't reset the stopwatch if you didn't notice (i just did ;p).

If you reset the SW the speed is actually faster when using a try/catch block, although I have no clue why is that. Example:

```
Code Stopwatch sw = new Stopwatch();
sw.Start();
```

```
if (checkBox1.Checked == true)
textBox1.Enabled = false;
```

```
Console.WriteLine(sw.Elapsed);
sw.Stop();
```

```
textBox1.Enabled = true;
```

```
Stopwatch sw2 = new Stopwatch();
sw2.Start();
```

```
try
{
if (checkBox1.Checked == true)
textBox1.Enabled = false;
}
```

```
catch { }
Console.WriteLine(sw2.Elapsed);
sw2.Stop();
```

ari12 - May 6, 2012, 10:12 am

The problem with this is that it changes the icon

Vuln - May 6, 2012, 11:23 am

(May 6, 2012 10:12 AM)ari12 Wrote: [\[CITE\]](#) The problem with this is that it changes the icon
You can change it back easily with resHacker (alternatively you can check out my binder source)
>> <http://www.angusj.com/resourcehacker>
>> <https://hackforums.net/showthread.php?tid=2459865>

IonU - May 7, 2012, 10:39 am

[The two golden rules of exception handling](#)

[Catching Exception is almost never justified and almost always harmful](#)

[Do NOT catch that exception!](#)

[InvalidOperationException: "Exception" in disguise? Or just the new ApplicationException?](#)

[Common mistakes when working with exceptions](#)

[The art of creating exceptions](#)

Please read.

Please understand.

Please accept.

Please do.

Thank you.

Romi - June 10, 2012, 5:52 am

nice share bro and thanks for sharing the sources. awesome binder and the GUI

ubanooba - June 10, 2012, 10:30 am

Wrapping statements in try-catch blocks just for the hell of it isn't good programming practice at all.

There is no chance of that code throwing an exception, and it just confuses the reader as to why it was there in the first place. If someone decides to modify the code, it's their own lookout for handling any errors that are thrown. There's a difference between error handling and "crash prevention".

gixslayer - June 10, 2012, 11:13 am

(Jun 10, 2012 10:30 AM)ubanooba Wrote: [\[CITE\]](#) Wrapping statements in try-catch blocks just for the hell of it isn't good programming practice at all.

There is no chance of that code throwing an exception, and it just confuses the reader as to why it was there in the first place. If someone decides to modify the code, it's their own lookout for handling any errors that are thrown. There's a difference between error handling and "crash prevention".

This. If someone is going into your application editing it at MSIL level (or simply copy + pasting the source generated from the MSIL) they should be the ones checking error handling. Why would you mess with that?

Just because it 'does no harm' doesn't mean you should. If you ever want to get into programming on a professional level writing clean and proper code is vital. Things like wrapping code that won't raise exceptions into try/catch statements is just a big no.

Chris White - September 13, 2014, 9:22 pm

The download link is invalid. The file got deleted. Please consider re-uploading the file.

SaketSaket - September 14, 2014, 11:38 am

(Sep 13, 2014 09:22 PM)Chris White Wrote: [\[CITE\]](#) The download link is invalid. The file got deleted. Please consider re-uploading the file.
This thread is 2 year old ;) Don't grave dig...

adrivanrex - January 16, 2015, 7:14 am

Anyone got the source code? I want to check the code I'm making a websocket type RAT now..

morphine1337 - January 16, 2015, 7:54 am

Check into codedom, it's much better to learn for yourself than from sources, IMO.

jenjente110 - November 27, 2016, 1:42 am

The Link is broken . Please reupload :)

gothic_coder - November 27, 2016, 4:29 am

broken link upload please

sososerious - November 27, 2016, 2:34 pm

Thanks for sharing. Im sure some of us were looking for such nice contribution.