How to use the try/catch block to catch exceptions

02/06/2019 • 2 minutes to read • +5

Place any code statements that might raise or throw an exception in a `try` block, and place statements used to handle the exception or exceptions in one or more `catch` blocks below the `try` block. Each `catch` block includes the exception type and can contain additional statements needed to handle that exception type.

In the following example, a StreamReader opens a file called *data.txt* and retrieves a line from the file. Since the code might throw any of three exceptions, it's placed in a `try` block. Three `catch` blocks catch the exceptions and handle them by displaying the results to the console.

C# Copy

```
using System;
using System.IO;

public class ProcessFile
{
    public static void Main()
    {
        try
        {
            using (StreamReader sr = File.OpenText("data.txt"))
            {
                Console.WriteLine($"The first line of this file is
{sr.ReadLine()}");
            }
        }
        catch (FileNotFoundException e)
        {
            Console.WriteLine($"The file was not found: '{e}'");
        }
        catch (DirectoryNotFoundException e)
        {
            Console.WriteLine($"The directory was not found: '{e}'");
        }
        catch (IOException e)
        {
            Console.WriteLine($"The file could not be opened: '{e}'");
        }
    }
}
```

The Common Language Runtime (CLR) catches exceptions not handled by `catch` blocks. If an exception is caught by the CLR, one of the following results may occur depending on your CLR configuration:

- A **Debug** dialog box appears.
- The program stops execution and a dialog box with exception information appears.
- An error prints out to the standard error output stream.

Note

Most code can throw an exception, and some exceptions, like **OutOfMemoryException**, can be thrown by the CLR itself at any time. While applications aren't required to deal with these exceptions, be aware of the possibility when writing libraries to be used by others. For suggestions on when to set code in a `try` block, see **Best Practices for Exceptions**.

See also

- Exceptions
- Handling I/O errors in .NET

Is this page helpful?

Yes No

Recommended content

How to convert a string to a number - C# Programming Guide

Learn how to convert a string to a number in C# by calling the Parse, TryParse, or Convert class methods.

try-catch - C# Reference

try-catch - C# Reference

How to concatenate multiple strings (C# Guide)

There are multiple ways to concatenate strings in C#. Learn the options and the reasons behind different choices.

NullReferenceException Class (System)

The exception that is thrown when there is an attempt to dereference a null object reference.

Show more