C# Try Catch – Exception Handling

C# Exception Handling

Exception handling is one of the most important aspect in any programming and for any developer. No matter how accurate the business logic is, there is always a chance that something could go out of our assumptions during program execution.

Some of the scenarios could be when the file that we would like to read is not available, or no privileges to read or write a file, or when trying to calculate a division and the denominator is zero, etc.

In this tutorial, we will how to handle these run-time exceptions in C# using Try-Catch.

C# Try Catch

C# Try Catch is used to execute a set of statements in try block which could potentially throw an Exception and then handle the exception using catch blocks.

Syntax of Try Catch

Following is the syntax of Try-Catch in C#.

```
try {
    // code that may throw an exception
 }
 catch(Exception ex) {
    // handle exception
 }
```

Try block contains set of statements (usually business logic).

Catch block contains set of statements about what to be done when an exception occurs. Like doing a work around or fail-safe or letting the user know about this exception.

Example 1 – C# Try Catch

Following is an example, where we read two numbers from user, and print the sum. We are converting each line provided by the user into integer using Convert.ToInt32() function.

If user provides an invalid number (say containing alphabets), Convert.ToInt32() throws an exception and if there is no code to catch the Exception, the program terminates. But, in here, we are using try-catch block to catch an exception if something goes wrong with the code in try block.

Program.cs

```
using System;

namespace CSharpExamples {

    class Program {
        static void Main(string[] args) {
            try {
                Console.Write("Enter number, a: ");
                int a = Convert.ToInt32(Console.ReadLine());

                Console.Write("Enter number, b: ");
                int b = Convert.ToInt32(Console.ReadLine());

                Console.WriteLine("a+b : " + (a + b));
            } catch(Exception ex){
                Console.WriteLine(ex.Message);
            }
            Console.WriteLine("Execution after try-catch block continues.");
        }
    }
}
```

Output

```
PS D:\workspace\csharp\HelloWorld> dotnet run
Enter number, a: 25
Enter number, b: 14
a+b : 39
Execution after try-catch block continues.
PS D:\workspace\csharp\HelloWorld> dotnet run
Enter number, a: 4asd
Input string was not in a correct format.
Execution after try-catch block continues.
```

During the first run, we have provided numbers in correct format. But in the second run, we provided input as `4asd` for a number. This throws an exception. We catch the exception, printed the error message to the user and continued with the execution of program statements after the try-catch block.

Why use Try Catch?

If an exception thrown by the code is not handled, it could terminate the program abruptly after the exception.

There are different types of Exceptions in C# that could be thrown. Many system functions that we use throw an

There are different types of Exceptions in C# that could be thrown. Many system functions that we use throw an exception. A developer should handle those exceptions considering all the fail-over scenarios for each C# statement in the program.

C# Errors [Solved]

✦ C# Error: Class does not contain a constructor that takes arguments