Online Training
- C#.NET Online Training Program
- ASP.NET Core Training
- Microservices Online Training using .NET Core
- Microsoft Azure Training

Introduction & Environment Setup
- How Computer Works
- Introduction to Programming Languages
- How Computer Programs Works
- Different Types of Applications
- Programming Methodologies
- Algorithm, Pseudocode, Programs, and Flowcharts
- Introduction to .NET Framework
- .NET Framework Architecture and Components
- Introduction to C# Programming Language
- How to Download and Install Visual Studio on Windows
- Creating First Console Application using Visual Studio
- .NET Developer Roadmap for 2025
- Coding Standard Best Practices

C#.NET Basics
- Basic Structure of C# Program
- Methods and Properties of Console Class in C#
- Data Types in C#
- Literals in C#
- Type Casting in C#
- Variables in C#
- Operators in C#
- Control Flow Statements in C#
- If-Else Statements in C#
- Switch Statements in C#
- Loops in C#
- While Loop in C#
- Do While Loop in C#
- For Loop in C#
- Break Statement in C#
- Continue Statement in C#
- Goto Statement in C#
- Functions in C#
- User-Defined Functions in C#
- Call By Value and Call By Reference in C#
- Recursion in C#
- User Input and Output in C#
- Command Line Arguments in C#
- String in C#
- Static Keyword in C#

Multiple Catch Blocks in C#

Back to: C#.NET Tutorials For Beginners and Professionals

Multiple Catch Blocks in C# with Examples

In this article, I am going to discuss how to implement **Multiple Catch Blocks in C#** to handle different types of exceptions for a single try block with examples. Please read our previous article before proceeding to this article where we discussed the basics of **Exception Handling in C#** with examples. As part of this article, I am going to discuss the following pointers.

1. **How to implement Multiple Catch Blocks in C#?**
2. **Is it possible to catch all exceptions using a single catch block?**
3. **When should we write multiple catch blocks in C# for a single try block?**

How to Implement Multiple Catch Blocks in C#?

It is possible in C#, to write multiple catch blocks for a given try block. When we implement multiple catch blocks in C# for a given try block, then at any given point of time only one catch block is going to be executed and other catch blocks will be ignored. With this kept in mind, let us proceed and see an example of how to implement Multiple Catch Blocks in C#.

Example to Understand Multiple Catch Blocks in C#

Let us see an example and try to understand how to implement multiple catch blocks for a given try block in C# and also try to understand the execution flow. Please have a look at the following example. As you can see, here, we created two catch blocks for the given try block. The first catch block takes the DivideByZeroException class as the input parameter and the second catch block takes the FormatException class as the input parameter.

```
using System;
namespace ExceptionHandlingDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            int Number1, Number2, Result;
            try
            {
                Console.WriteLine("Enter First Number");
                Number1 = int.Parse(Console.ReadLine());
                Console.WriteLine("Enter Second Number");
                Number2 = int.Parse(Console.ReadLine());
                Result = Number1 / Number2;
                Console.WriteLine($"Result: {Result}");
            }
            catch (DivideByZeroException DBZE)
            {
                Console.WriteLine("Second Number Should Not Be Zero");
            }
            catch (FormatException FE)
            {
                Console.WriteLine("Enter Only Integer Numbers");
            }
            Console.ReadKey();
        }
    }
}
```

Run the above application and entered values like 10, and 0. It will give you the below output. In this case, when we entered the second number as 0, the CLR implicitly throws the **DivideByZeroException** instance and the catch block which takes as a parameter of DivideByZeroException will handle the exception, and hence the message Second Number Should Not Be Zero will be printed on the console as shown in the below image.

```
Enter First Number
10
Enter Second Number
0
Second Number Should Not Be Zero
```

Again, run the application and entered the values as 10 and abc. This time it will give you the following output. In this case, when we entered the second value as abc, the CLR implicitly throws **FormatException** instance and the catch block which takes as a parameter of FormatException will handle the exception and hence, the message Enter Only Integer Numbers will be printed on the console as shown in the below image.

Static vs Non-Static Members in C#
Const and Read-Only in C#
Properties in C#
Why we Should Override ToString Method in C#
Override Equals Method in C#
Difference Between Convert.ToString and ToString Method in c#
Checked and Unchecked Key words in C#
Stack and Heap Memory in .NET
Boxing and Unboxing in C#

OOPs in C#
Object Oriented Programming (OOPs) in C#
Class and Objects in C#
Constructors in C#
Types of Constructors in C#
Why We Need Constructors in C#
Static vs Non-Static Constructors in C#
Private Constructors in C#
Destructors in C#
Garbage Collection in .NET Framework
Differences Between Finalize and Dispose in C#
Access Specifiers in C#
Encapsulation in C#
Abstraction in C#
Inheritance in C#
Types of Inheritance in C#
How to use Inheritance in Application Development
IsA and HasA Relationship in C#
Generalization and Specialization in C#
Abstract Class and Abstract Methods in C#
Abstract Class and Abstract Methods Interview Questions in C#
How to Use Abstract Classes and Methods in C# Application
Interface in C#
Interface Interview Questions and Answers in C#
Interface Realtime Examples in C#
Multiple Inheritance in C#
Multiple Inheritance Realtime Example in C#
Polymorphism in C#
Method Overloading in C#
Operator Overloading in C#
Method Overriding in C#
Method Hiding in C#
Partial Class and Partial Methods in C#
Sealed Class and Sealed Methods in C#

```
Enter First Number
10
Enter Second Number
abc
Enter Only Integer Numbers
```

Whenever we implement Multiple Catch Blocks in C#, then it is not possible to write the catch blocks in the following manner, it raises to compilation error because the first catch block Exception can handle all the exceptions as it is the parent class of the child exception classes and it does not make any sense to write the further catch blocks as they are never going to be executed.

```
try
{
   //Code
}
catch (Exception ex)
{
   //Code
}
catch (FormatException fex)
{
   //Code
}
```

Is it possible to catch all exceptions using a single catch block in C#?

Yes, it is possible. We can catch all exceptions with a single catch block with the parameter Exception. The Exception class is the superclass of all Child Exception classes and hence it can handle all types of exceptions thrown in the try block. We need to use this catch block only for stopping the abnormal termination irrespective of the exceptions thrown from its corresponding try block. For a better understanding, please have a look at the following example.

```
using System;
namespace ExceptionHandlingDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            int Number1, Number2, Result;
            try
            {
                Console.WriteLine("Enter First Number");
                Number1 = int.Parse(Console.ReadLine());
                Console.WriteLine("Enter Second Number");
                Number2 = int.Parse(Console.ReadLine());
                Result = Number1 / Number2;
                Console.WriteLine($"Result: {Result}");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Generic Catch Block...");
            }
            Console.ReadKey();
        }
    }
}
```

Now, in the above example, for any type of exception, the generic catch block is going to be executed. It is always recommended to write a catch block with the Exception parameter as the last catch block even though we are writing multiple catch blocks. It acts as a backup catch block. Following is the syntax to do the same.

```
try
{
   //Code
}
catch(DivideByZeroException dbe)
{
   //Code
}
catch (FormatException fe)
{
   //Code
}
catch (Exception e)
{
   //Code
}
```

In the below example, we are writing the generic catch block as the last catch block and if any exception thrown in the try block is not handled by any specific catch block, then the generic block is going to be executed.

- Extension Methods in C#
- Static Class in C#
- Variable Reference and Instance of a Class in C#

OOPs Real-Time Examples
- Real-time Examples of Encapsulation Principle in C#
- Real-Time Examples of Abstraction Principle in C#
- Real-Time Examples of Inheritance Principle in C#
- Real-Time Examples of Polymorphism Principle in C#
- Real-Time Examples of Interface in C#
- Real-Time Examples of Abstract Class in C#

Exception Handling
- Exception Handling in C#
- **Multiple Catch Blocks in C#**
- Finally Block in C#
- How to Create Custom Exceptions in C#
- Inner Exception in C#
- Exception Handling Abuse in C#

Events, Delegates and Lambda Expression in C#
- Course Structure of Events, Delegates and Lambda Expression
- Roles of Events, Delegates and Event Handler in C#
- Delegates in C#
- Multicast Delegates in C#
- Delegates Real-Time Example in C#
- Generic Delegates in C#
- Anonymous Method in C#
- Lambda Expressions in C#
- Events in C# with Examples

Multi-Threading
- Multithreading in C#
- Thread class in C#
- How to Pass Data to Thread Function in Type Safe Manner in C#
- How to Retrieve Data from a Thread Function in C#
- Join Method and IsAlive Property of Thread Class in C#
- Thread Synchronization in C#
- Lock in C#
- Monitor Class in C#
- Mutex Class in C#
- Semaphore Class in C#
- SemaphoreSlim Class in C#
- Deadlock in C#
- Performance Testing of a Multithreaded Application
- Thread Pool in C#
- Foreground and Background Threads in C#
- AutoResetEvent and ManualResetEvent in C#
- Thread Life Cycle in C#
- Threads Priorities in C#
- How to Terminate a Thread in


```
using System;
namespace ExceptionHandlingDemo
{
    class Program
    {
        static void Main(string[] args)
        {
            int Number1, Number2, Result;
            try
            {
                Console.WriteLine("Enter First Number");
                Number1 = int.Parse(Console.ReadLine());
                Console.WriteLine("Enter Second Number");
                Number2 = int.Parse(Console.ReadLine());
                Result = Number1 / Number2;
                Console.WriteLine($"Result: {Result}");
            }

            catch (DivideByZeroException DBZE)
            {
                Console.WriteLine("Second Number Should Not Be Zero");
            }
            catch (FormatException FE)
            {
                Console.WriteLine("Enter Only Integer Numbers");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Generic Catch Block...");
            }
            Console.ReadKey();
        }
    }
}
```

Now, run the above application and entered big integer values like 23456789234. It will give you the following output. In this case, when we entered the number 23456789234, the CLR implicitly throw the OverflowException instance and we don't have any catch block which takes as a parameter of OverflowException, hence the generic catch block is going to execute and display the following message.

```
Enter First Number
23456789234
Generic Catch Block...
```

When should we write Multiple Catch Blocks in C# for a Single Try block?

We need to write multiple catch blocks in C# for a single try block because of the following reasons

1. To print messages specific to an exception or
2. To execute some logic specific to an exception

In the next article, I am going to discuss how to create a **Finally Block in C#** with Examples. Here, in this article, I try to explain how to implement **Multiple Catch Blocks in C#** with Examples. I hope you understood this as well as enjoy this Multiple Catch Blocks C# with Examples article.

DOT NET TUTORIALS

Dot Net Tutorials

About the Author: Pranaya Rout

Pranaya Rout has published more than 3,000 articles in his 11-year career. Pranaya Rout has very good experience with Microsoft Technologies, Including C#, VB, ASP.NET MVC, ASP.NET Web API, EF, EF Core, ADO.NET, LINQ, SQL Server, MYSQL, Oracle, ASP.NET Core, Cloud Computing, Microservices, Design Patterns and still learning new technologies.

Previous Lesson
Exception Handling in C#

Next Lesson
Finally Block in C#

2 thoughts on "Multiple Catch Blocks in C#"

JAMES
AUGUST 13, 2019 AT 6:33 PM

C#
Inter Thread Communication in C#
How to Debug a Multi-threaded Application in C#

Collections in C#
Arrays in C#
2D Arrays in C#
Advantages and Disadvantages of Arrays in C#
Collections in C#
ArrayList in C#
Hashtable in C#
Non-Generic Stack in C#
Non-Generic Queue in C#
Non-Generic SortedList in C#
Advantages and Disadvantages of Non-Generic Collection in C#
Generic Collections in C#
Generics in C#
Generic Constraints in C#
Generic List Collection in C#
How to Sort a List of Complex Type in C#
Comparison Delegate in C#
Dictionary Collection Class in C#
Conversion Between Array List and Dictionary in C#
List vs Dictionary in C#
Generic Stack Collection Class in C#
Generic Queue Collection Class in C#
Foreach Loop in C#
Generic HashSet Collection Class in C#
Generic SortedList Collection Class in C#
Generic SortedSet Collection Class in C#
Generic SortedDictionary Collection Class in C#
Generic LinkedList Collection Class in C#
Concurrent Collection in C#
ConcurrentDictionary Collection Class in C#
ConcurrentQueue Collection Class in C#
ConcurrentStack Collection Class in C#
ConcurrentBag Collection Class in C#
BlockingCollection in C#

File Handling
File Handling in C#
FileStream Class in C#
StreamReader and StreamWriter in C#
File Class in C#
TextWriter and TextReader in C#
BinaryWriter and BinaryReader in C#
StringWriter and StringReader in C#
FileInfo Class in C#
DirectoryInfo Class in C#
Export and Import Excel Data

Best tutorials to learn C#.net.
Thank you

Reply

ABHI
JANUARY 6, 2021 AT 9:16 PM

Very concise and god examples. Great job.

Reply

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name* Email* Website

Post Comment

in C#

Asynchronous Programming
- Introduction to Concurrency
- Async and Await in C#
- Task in C#
- How to Return a Value from Task in C#
- How to Execute Multiple Tasks in C#
- How to Limit Number of Concurrent Tasks in C#
- How to Cancel a Task in C# using Cancellation Token
- How to Create Synchronous Method using Task in C#
- Retry Pattern in C#
- Only One Pattern in C#
- How to Control the Result of a Task in C#
- Task-Based Asynchronous Programming in C#
- Chaining Tasks by Using Continuation Tasks
- How to Attached Child Tasks to a Parent Task in C#
- ValueTask in C#
- How to Cancel a Non-Cancellable Task in C#
- Asynchronous Streams in C#
- How to Cancel Asynchronous Stream in C#

Parallel Programming
- Task Parallel Library in C#
- Parallel For in C#
- Parallel Foreach Loop in C#
- Parallel Invoke in C#
- Maximum Degree of Parallelism in C#
- How to Cancel Parallel Operations in C#
- Atomic Methods Thread Safety and Race Conditions in C#
- Interlocked vs Lock in C#
- Parallel LINQ in C#
- Multithreading vs Asynchronous Programming vs Parallel Programming in C#

AutoMapper
- AutoMapper in C#
- AutoMapper Complex Mapping in C#
- How to Map Complex Type to Primitive Type using AutoMapper in C#
- AutoMapper Reverse Mapping in C#
- AutoMapper Conditional Mapping in C#
- AutoMapper Ignore Method in C#
- Fixed and Dynamic Values in Destination Property in AutoMapper

Optional Parameter, Indexers and Enums
- How to make Optional Parameters in C#
- Indexers in C#
- Indexers Real-Time Example in C#
Enums in C#

.NET Framework Architecture
DOT NET Framework
Common Language Runtime in .NET Framework
.NET Program Execution Process
Intermediate Language (ILDASM & ILASM) Code in C#
Common Type System in .NET Framework
Common Language Specification in .NET Framework
Managed and Unmanaged Code in .NET Framework
Assembly DLL EXE in .NET Framework
App Domain in .NET Framework
Strong and Weak Assemblies in .NET Framework
How to Install an Assembly into GAC in .NET Framework
DLL Hell Problem and Solution in .NET Framework

Var, Dynamic and Reflection
Reflection in C#
Dynamic Type in C#
Var Keyword in C#
Var vs Dynamic in C#
Dynamic vs Reflection in C#
Volatile Keyword in C#
Ref vs Out in C#
Named Parameters in C#

C# 7.X new Features
C# 7 New Features
Enhancement in Out Variables in C# 7
Pattern Matching in C#
Digit Separators in C# 7
Tuples in C# 7
Splitting Tuples in C# 7
Local Functions in C# 7
Ref Returns and Ref Locals in C# 7
Generalized Async Return Types in C# 7
Expression Bodied Members in C#
Thrown Expression in C#
Async Main in C#

C# 8 New Features
C# 8 New Features
ReadOnly Structs in C#
Default Interface Methods in C#
Pattern Matching in C#
Using Declarations in C#
Static Local Functions in C#
Disposable Ref Structs in C#
Nullable Reference Types in C# 8
Asynchronous Streams in C#
Asynchronous Disposable in C#
Indices and Ranges in C#

- Null-Coalescing Assignment Operator in C#
- Unmanaged Constructed Types in C#
- Stackalloc in in C#

Most Popular C# Books

- Most Recommended C# Books
- Most Recommended Data Structure and Algorithms Books using C#

© Dot Net Tutorials | Website Design by Sunrise Pixel