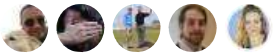
How to use specific exceptions in a catch block

03/30/2017 • 2 minutes to read •  +5

In this article

In general, it's good programming practice to catch a specific type of exception rather than use a basic `catch` statement.

When an exception occurs, it is passed up the stack and each catch block is given the opportunity to handle it. The order of catch statements is important. Put catch blocks targeted to specific exceptions before a general exception catch block or the compiler might issue an error. The proper catch block is determined by matching the type of the exception to the name of the exception specified in the catch block. If there is no specific catch block, the exception is caught by a general catch block, if one exists.

The following code example uses a `try/catch` block to catch an InvalidCastException. The sample creates a class called `Employee` with a single property, employee level (`Emlevel`). A method, `PromoteEmployee`, takes an object and increments the employee level. An InvalidCastException occurs when a DateTime instance is passed to the `PromoteEmployee` method.

C# Copy

```
using System;

public class Employee
{
    //Create employee level property.
    public int Emlevel
    {
        get
        {
            return(emlevel);
        }
        set
        {
            emlevel = value;
        }
    }

    private int emlevel = 0;
}
```

```
public class Ex13
{
    public static void PromoteEmployee(Object emp)
    {
        // Cast object to Employee.
        var e = (Employee) emp;
        // Increment employee level.
        e.Emlevel = e.Emlevel + 1;
    }

    public static void Main()
    {
        try
        {
            Object o = new Employee();
            DateTime newYears = new DateTime(2001, 1, 1);
            // Promote the new employee.
            PromoteEmployee(o);
            // Promote DateTime; results in InvalidCastException as newYears
is not an employee instance.
            PromoteEmployee(newYears);
        }
        catch (InvalidCastException e)
        {
            Console.WriteLine("Error passing data to PromoteEmployee method.
" + e.Message);
        }
    }
}
```

See also

- Exceptions

Is this page helpful?

Yes No

Recommended content

try-catch-finally - C# Reference

try-catch-finally - C# Reference

Exceptions and Exception Handling - C# Programming Guide

Learn about exceptions and exception handling. These C# features help deal with unexpected or exceptional situations that happen when a program is running.

How to: Explicitly Throw Exceptions

Learn how to throw an exception explicitly in .NET by using the C# throw statement or the Visual Basic Throw statement.

Creating and Throwing Exceptions

Learn about creating and throwing exceptions. Exceptions are used to indicate that an error has occurred while running a program.

Show more