Catching, Throwing, and Rethrowing Exceptions

Jason Roberts
.NET DEVELOPER

@robertsjason   dontcodetired.com

Overview

Throwing exceptions from expressions

Catching different exception types with multiple catch blocks

Understanding the finally block

Rethrowing exceptions and preserving the stack trace

Catching and wrapping exceptions

Filtering catch blocks with exception filters

Global unhandled exception handling

Switch expression

“The switch expression provides for switch-like semantics in an expression context. It provides a concise syntax when the switch arms produce a value.”

https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/switch-expression

```
public class Calculator
{
    public int Calculate(int number1, int number2,
                         string operation) => operation switch
    {
        "/" => Divide(number1, number2),
        "+" => Add(number1, number2),

    };

    private int Divide(int number, int divisor)
                                   => number / divisor;
    private int Add(int number1, int number2)
                                   => number1 + number2;
}
```

```
public class Calculator
{
    public int Calculate(int number1, int number2,
                     string operation) => operation switch
    {
        "/" => Divide(number1, number2),
        "+" => Add(number1, number2),
         _  => throw new ArgumentOutOfRangeException()
    };

    private int Divide(int number, int divisor)
                                => number / divisor;
    private int Add(int number1, int number2)
                                => number1 + number2;
}
```

Summary

Throwing exceptions from expressions

Catching different exception types with multiple catch blocks

Understanding the finally block

throw ex;   throw;

Catching and wrapping exceptions

- Inner exception

when (ex.ParamName == "operation")

currentAppDomain.UnhandledException

Up Next:

Creating and Using Custom Exceptions