Getting Started with Exceptions

Jason Roberts
.NET DEVELOPER

@robertsjason dontcodetired.com

Overview

Understanding exception handling

- Exception “bubbling”
- try...catch...finally

Demo code overview

Causing an exception

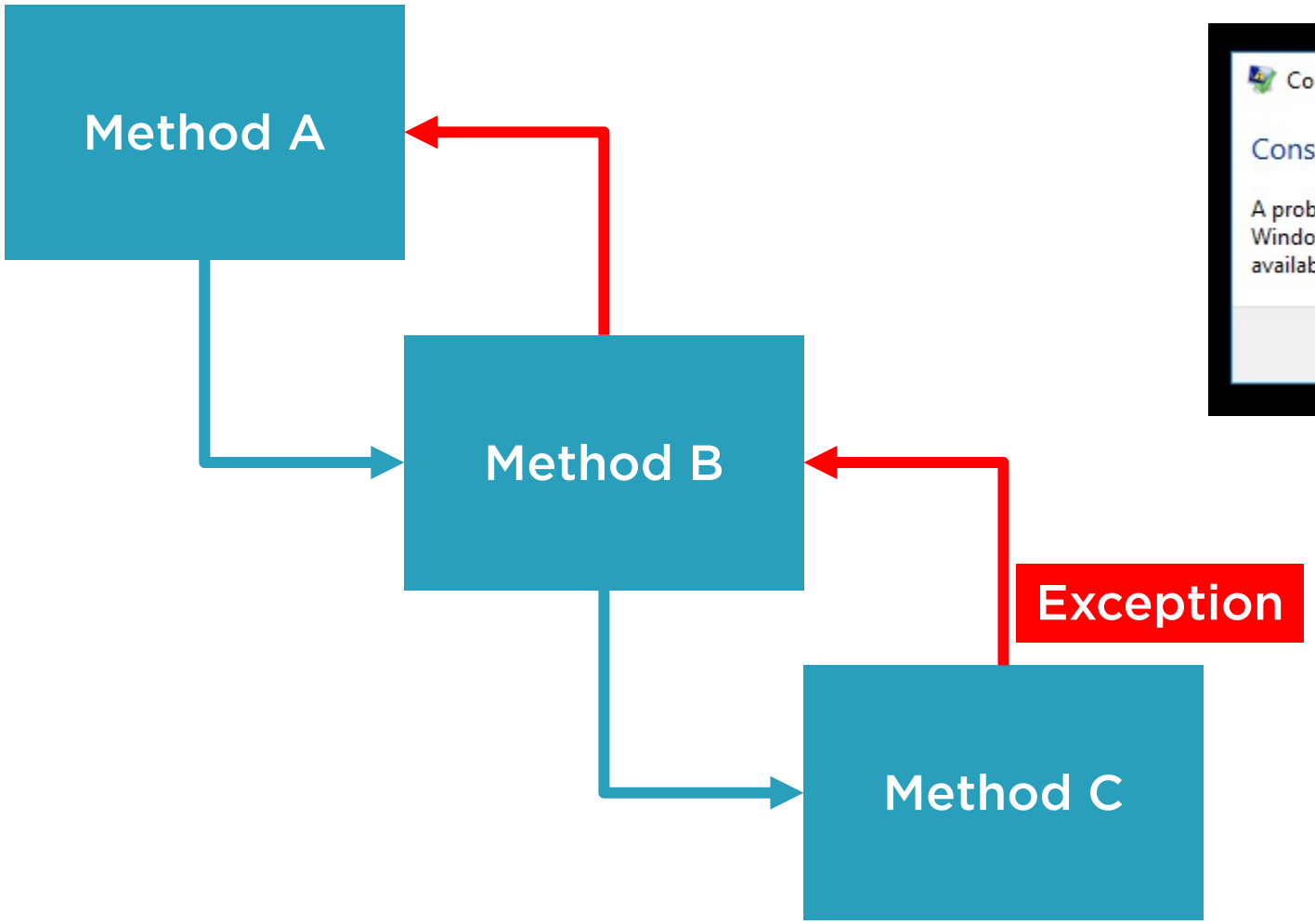
Understanding the stack trace

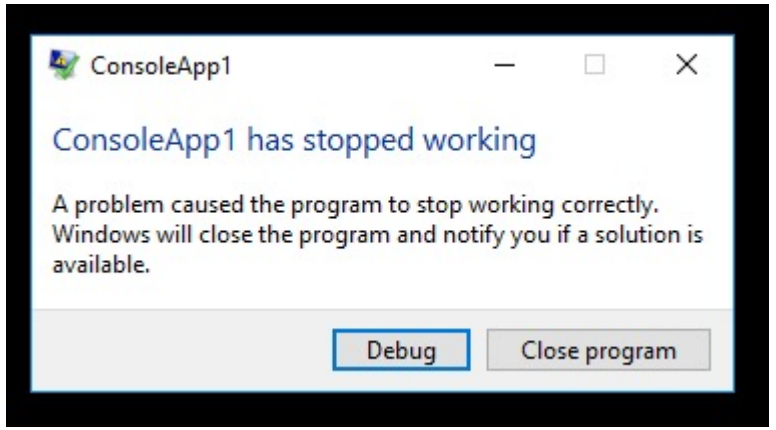
Creating and throwing an exception

Getting started with exception catching

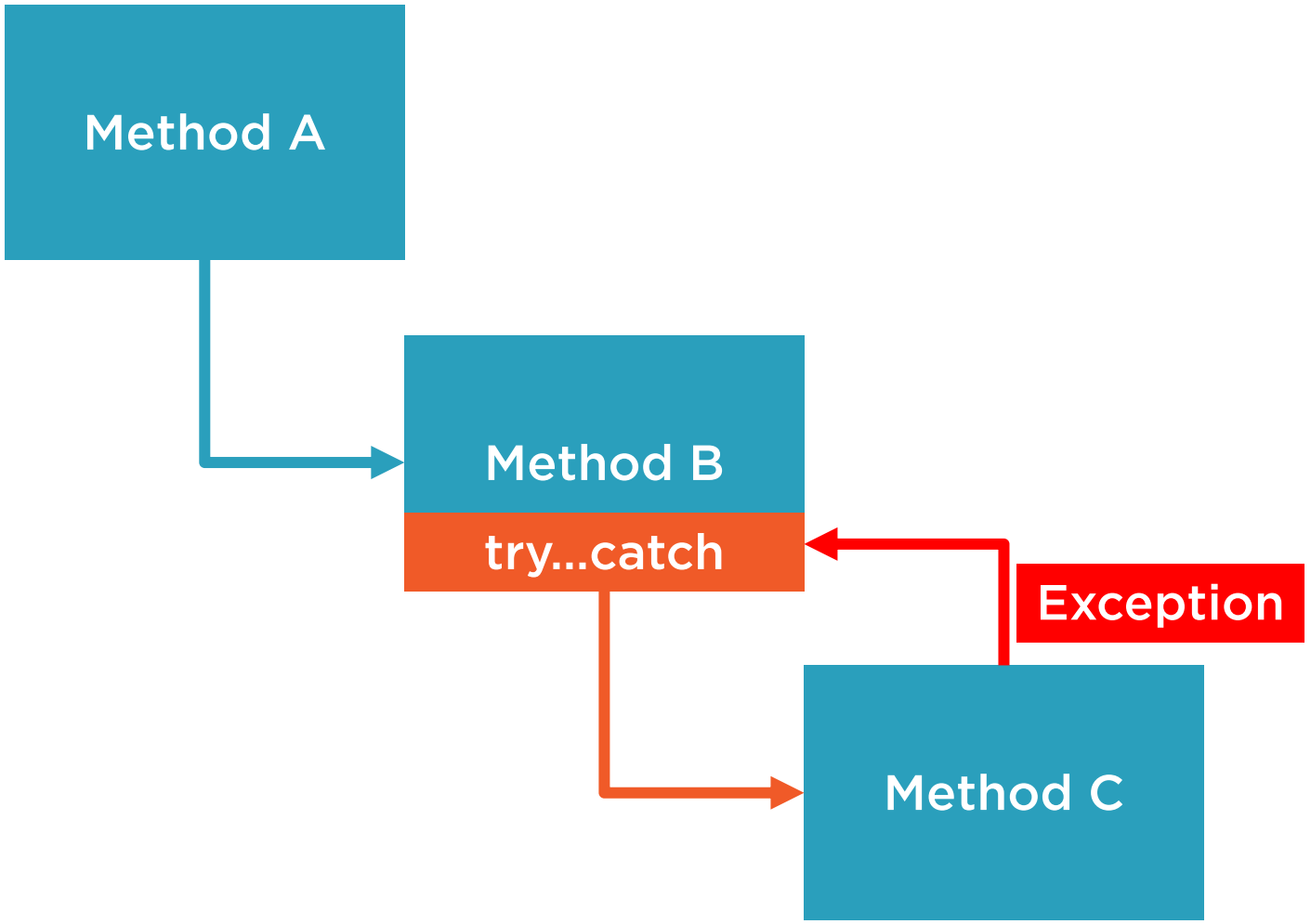
Exception handling good practices

Understanding Exception Handling





Understanding Exception Handling



Introducing the Try Statement

```
try
{
    // Some operation
}

catch (ArgumentNullException ex)
{
    // Handle ArgumentNullException
}

catch (InvalidOperationException ex)
{
    // Handle InvalidOperationException
}

catch (Exception ex)
{
    // Handle all other exceptions
}
```

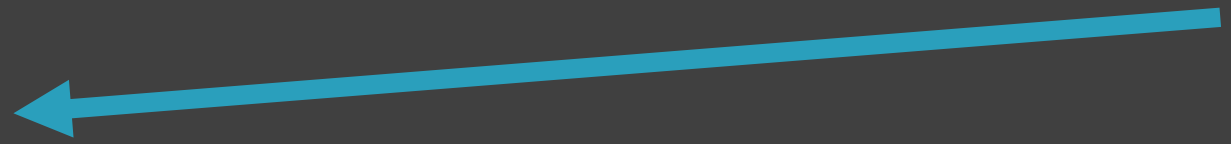
Introducing the Try Statement

```
try
{
    // Some operation
}

catch (ArgumentNullException ex)
{
    // Handle ArgumentNullException
}

catch (InvalidOperationException ex)
{
    // Handle InvalidOperationException
}

catch
{
    // No exception variable
}
```



Introducing the Try Statement

```
try
{
    // Some operation
}

finally
{
    // Always executed when control leaves try block
}
```

Introducing the Try Statement

```
try
{
    // Some operation
}

catch (ArgumentNullException ex)
{
    // Handle ArgumentNullException
}

finally
{
    // Always executed when control leaves try block
}
```

Exception Handling Good Practices

Do not add a catch block that does nothing or just rethrows

Catch block should add some value

May just be to log the error

Usually bad practice to ignore (swallow/trap) exceptions

Exception Handling Good Practices

Do not use exceptions for normal program flow logic

E.g. input validation

- You expect input to invalid sometimes
- Not an exceptional situation
- Part of expected logic flow

IsValid(xxx) method(s)

Exception Handling Good Practices

Design code to avoid exceptions

```
int Parse(string input)

bool TryParse(string input, out int result)

if (cn.State != ConnectionState.Closed)
{
   cn.Close();
}
```

Consider returning null (or null object pattern) for extremely common errors

Exception Handling Good Practices

Use correct grammar in exception messages

Correct punctuation

Correct spelling

End sentences with full stop

Consider error message localization

Exception Handling Good Practices

Use finally blocks for cleanup

E.g. calling Dispose()

Callers of method should be able to assume no unexpected side effects when exception thrown/caught

Summary

Understanding exception handling

- Exception “bubbling”
- try...catch...finally

Caused a DivideByZeroException

- Visual Studio debugger
- Windows error dialog

Stack trace (and other exception properties)

Threw ArgumentNullException

Catching exceptions

Exception handling good practices

Up Next:

Catching, Throwing, and Rethrowing Exceptions