

Understanding the Exception Class Hierarchy



Jason Roberts

.NET DEVELOPER

@robertsjason dontcodetired.com



Overview



What does an exception represent?

The exception class hierarchy

The `System.Exception` base class

Commonly used constructors

`System.ApplicationException` guidelines

Commonly encountered exceptions



Exception

An exception is any error condition or unexpected behavior that is encountered by an executing program.

Microsoft Documentation

<https://docs.microsoft.com/en-us/dotnet/standard/exceptions/>



System and Application Exceptions

System

Third party

Your code

.NET Runtime (CLR)

.NET Framework

OutOfMemory

StackOverflow

Libraries/frameworks

JsonSerialization

RulesEngine



The actual type of the exception class represents the kind of error that occurred.

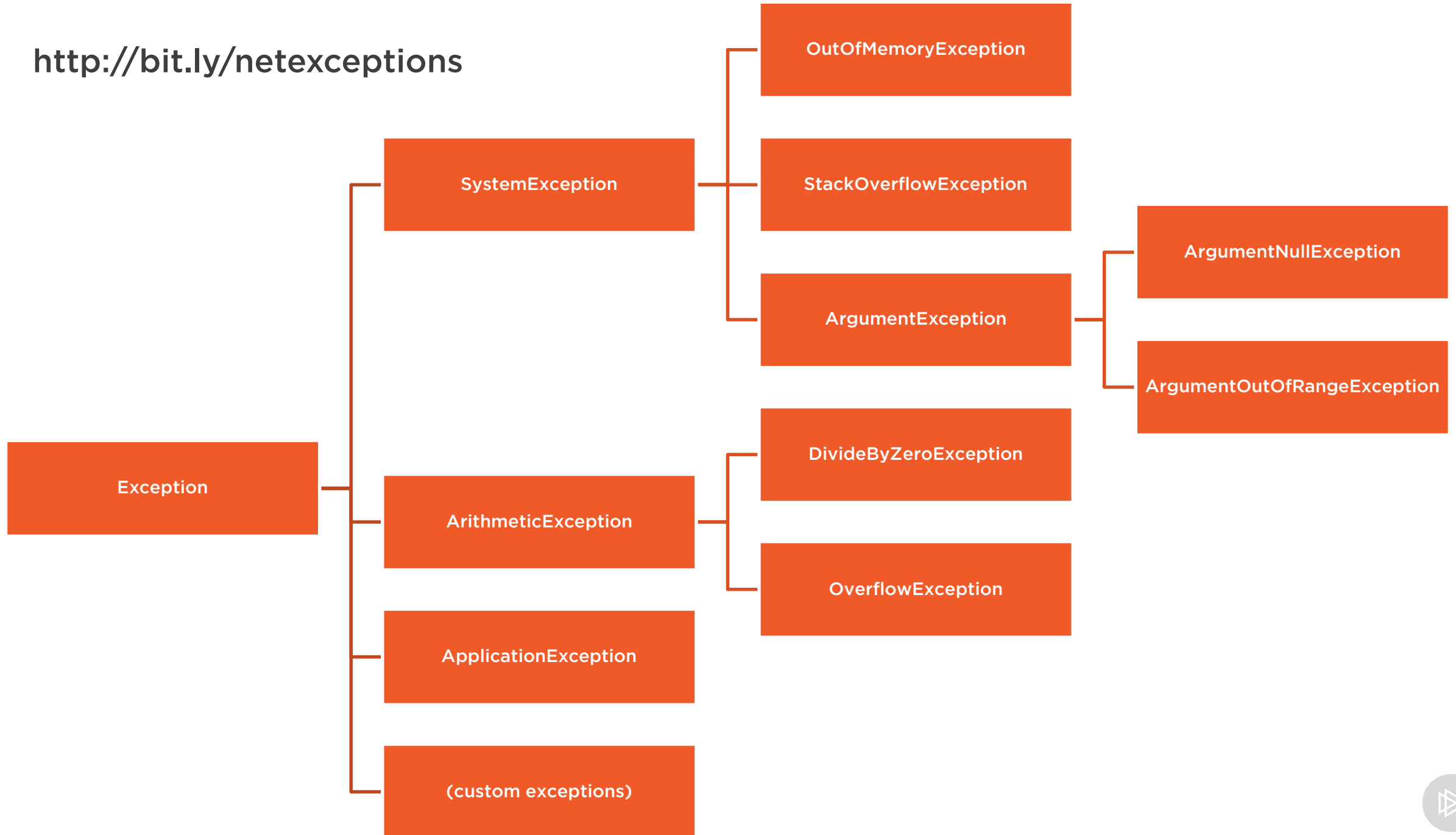
Any additional property values that are set help to further refine/define the error.



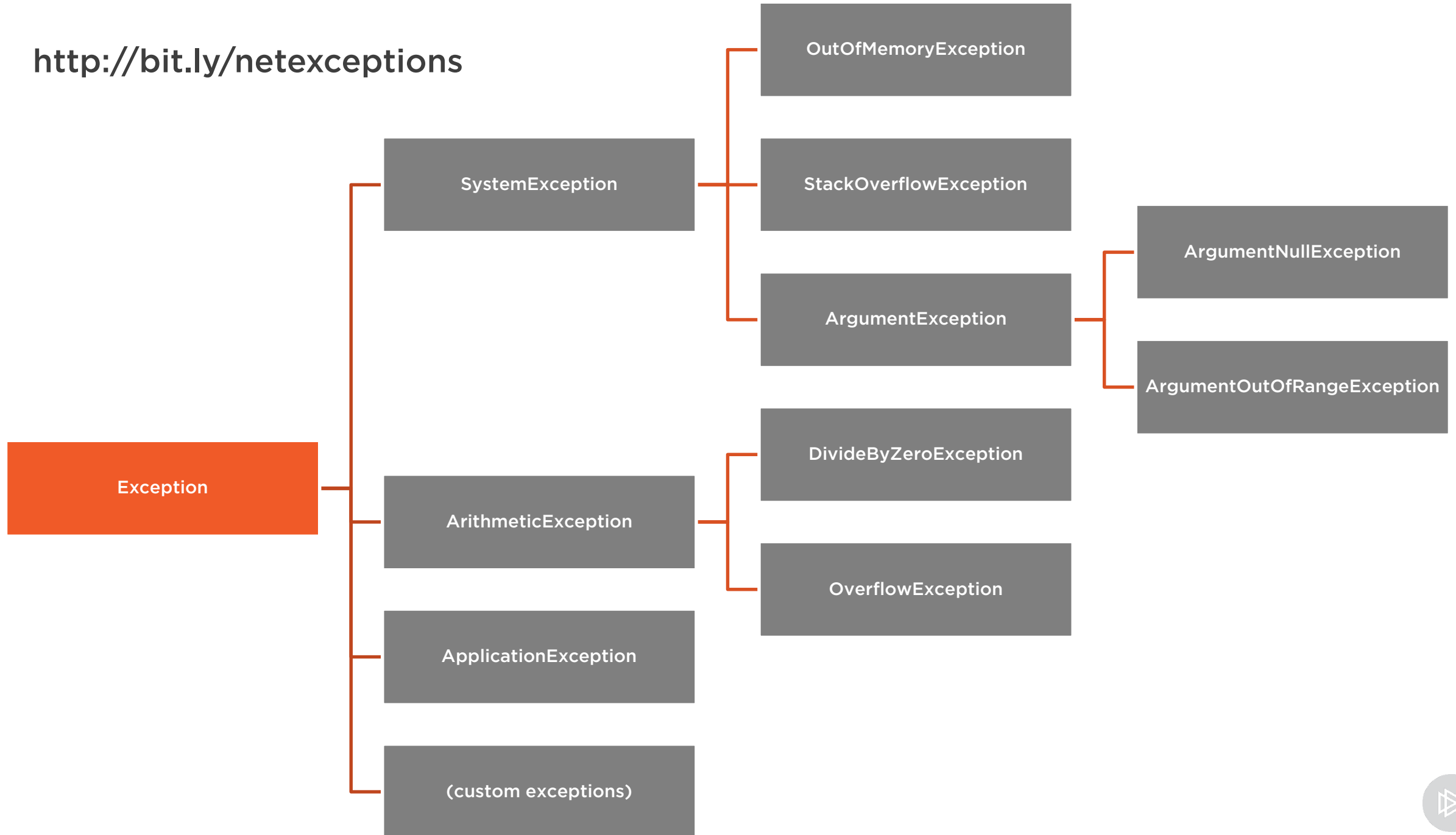
The Exception Class Hierarchy



<http://bit.ly/netexceptions>



<http://bit.ly/netexceptions>



The `System.Exception` class
is the base class for all
types of exceptions



Message
StackTrace
Data
InnerException
Source
HResult
HelpLink
TargetSite

System.Exception
Properties



Message

String

Describes the reason for the exception

Write for the developer who going to handling the exception

Should completely describe the error

Should describe how to correct error (where possible/applicable)

May sometimes be shown to end-user

May sometimes be logged

Correct grammar

Don't include passwords/security/sensitive data



StackTrace

String

Information about call stack

Trace of the method calls leading to exception

Helps to show the execution path/flow that led to exception



Data

Dictionary

Key/value pairs

String key

Object value

Arbitrary number of items

Additional/supplementary user-defined
exception data

Don't include passwords/security/sensitive
data in keys/values

Be careful of key conflicts



InnerException

System.Exception

Capture the preceding exception in new exception

Exception “wrapping”



Source

String

Application/object name that caused error

Defaults to name of originating assembly



HResult

Int32

Represents a HRESULT numerical value

Often used with COM-interop



HelpLink

String

Link to associated help file

Uniform Resource Name (URN)

Uniform Resource Locator (URL)



TargetSite

System.Reflection.MethodBase

Method that threw current exception

- Name
- Return type
- Is public/private
- Etc.



```
public Exception()
```

```
public Exception(  
    string message  
)
```

```
public Exception(  
    string message,  
    Exception innerException  
)
```

◀ Default Message property and null InnerException

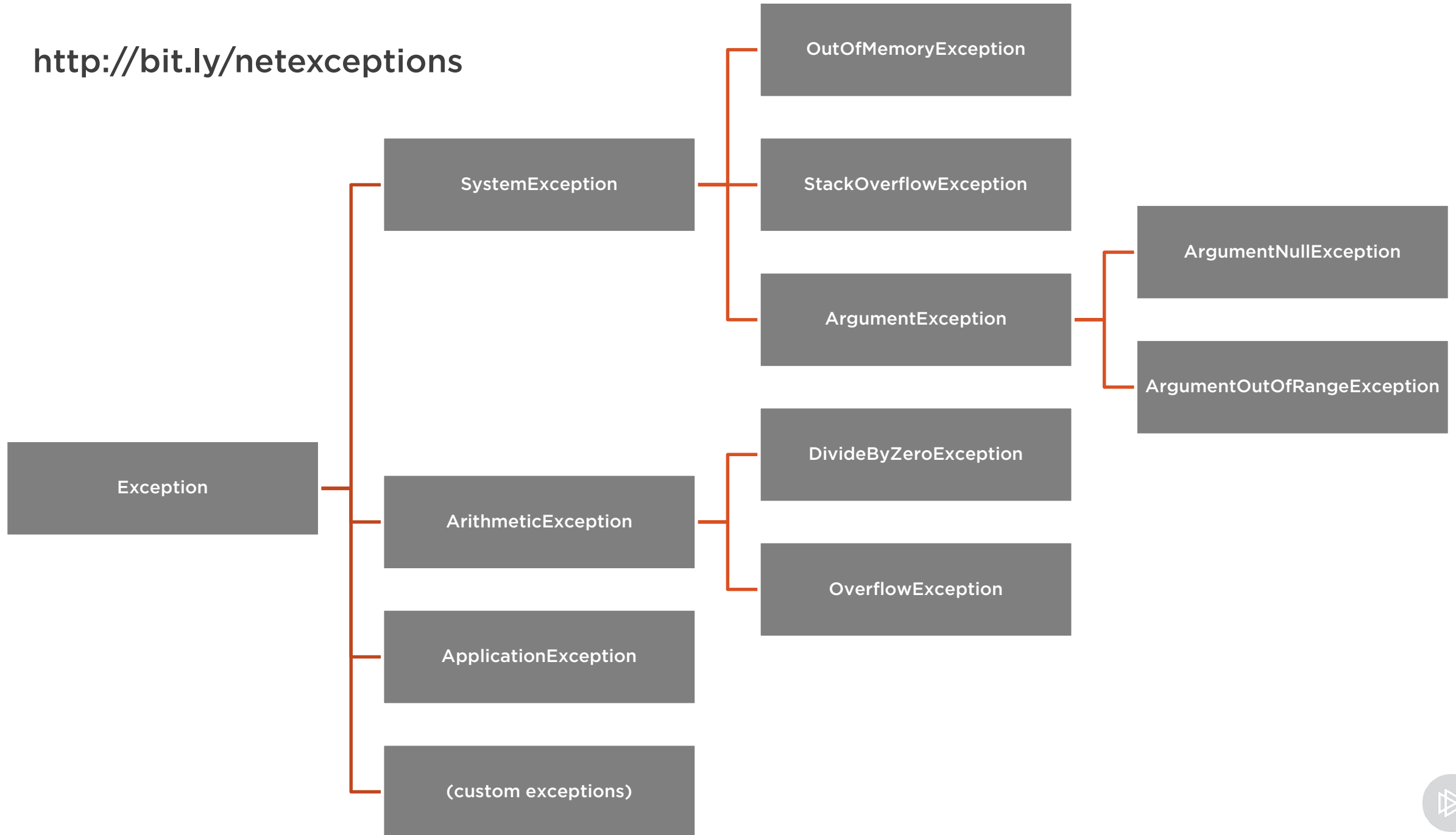
◀ User defined Message

◀ User defined Message

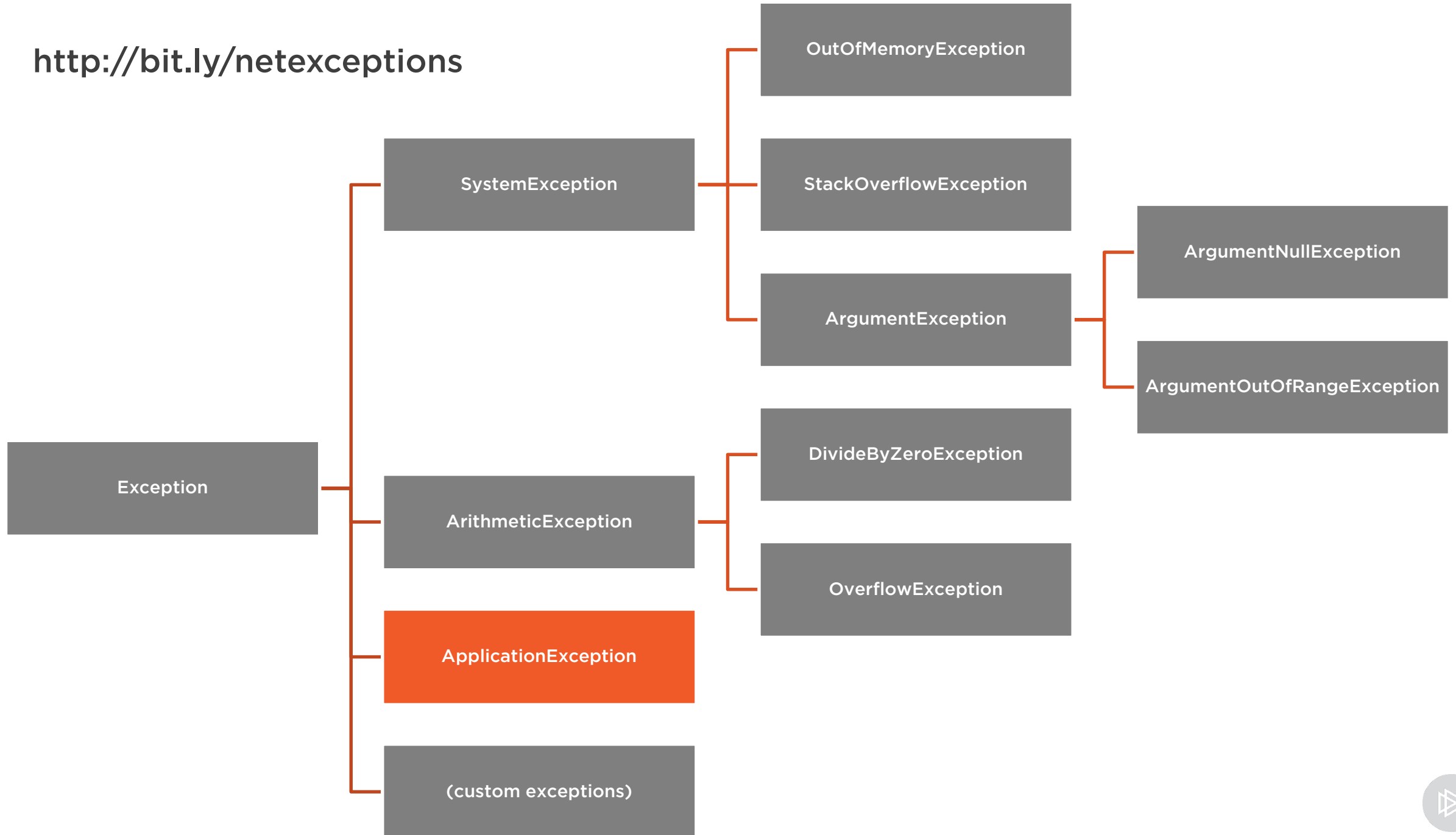
◀ Wrapped exception



<http://bit.ly/netexceptions>



<http://bit.ly/netexceptions>



“System.ApplicationException is a class that should not be part of the .NET Framework. The original idea was that classes derived from SystemException would indicate exceptions thrown from the CLR (or system) itself, whereas non-CLR exceptions would be derived from ApplicationException. However, a lot of exception classes didn't follow this pattern.”

Framework Design Guidelines

<https://blogs.msdn.microsoft.com/kcwalina/2006/06/23/applicationexception-considered-useless/>



ApplicationException Guidelines



An ApplicationException should not be thrown by your code



An ApplicationException exception should not be caught (unless you rethrow the original exception)



Custom exceptions should not be derived from ApplicationException

Commonly Encountered Exceptions

Exception &
SystemException

Exception: Represents execution errors

SystemException: Base class for exceptions in system exceptions namespace

Do not throw

Do not catch (except in top-level handlers)

Do not catch in framework code (unless rethrowing)



Commonly Encountered Exceptions

InvalidOperationException

Thrown when the current state of the object is invalid for a specific method being called

Throw when your object is in an inappropriate state when a method is called



Commonly Encountered Exceptions

ArgumentException,
ArgumentNullException, &
ArgumentOutOfRangeException

ArgumentException: Thrown when a method argument is invalid. (base class)

ArgumentNullException: Thrown when a null is passed to a method argument and it cannot accept nulls

ArgumentOutOfRangeException: Thrown when a method argument is outside of an allowable range

Prefer the most specific derived exception

Set the ParamName property when throwing one of the subclasses of ArgumentException



Commonly Encountered Exceptions

NullPointerException &
IndexOutOfRangeException

NullPointerException: Thrown when an attempt is made to dereference a null object reference

IndexOutOfRangeException: Thrown when attempting to access an array/collection item that is outside its bounds

Reserved for runtime use

Usually indicate a bug in the program

Do not throw

Check arguments to avoid



Commonly Encountered Exceptions

StackOverflowException

Thrown when too many nested method calls cause the execution stack to overflow

Reserved and thrown by runtime

Do not explicitly throw

Do not catch StackOverflowException

Usually impossible to correct



Commonly Encountered Exceptions

OutOfMemoryException

Thrown when there is not enough memory to continue executing the program

Reserved and thrown by runtime

Do not explicitly throw

"If you choose to handle the exception, you should include a catch block that calls the `Environment.FailFast` method to terminate your app and add an entry to the system event log" - <http://bit.ly/outofmemory>



Summary



Exceptions represent any error condition or unexpected behaviour

Exceptions are organized in a hierarchy

System.Exception base class

System.Exception constructors

- public Exception()
- public Exception(string message)
- public Exception(string message, Exception innerException)

System.ApplicationException guidelines

Commonly encountered exceptions

- ArgumentException
- NullReferenceException



Up Next:

Getting Started with Exceptions

