System.Exception Class

[ILASM]

```
.class public serializable Exception extends System.Object
```

[C#]

```
public class Exception
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 1.0.x.x
- *Attributes:*
    - CLSCompliantAttribute(true)

Summary

Represents errors that occur during application execution.

Inherits From: System.Object

Library: BCL

Thread Safety: All public static members of this type are safe for multithreaded operations. No instance members are guaranteed to be thread safe.

Description

This class is the base class for all Exceptions.

When an error occurs, either the system or the currently executing application reports it by throwing an Exception containing information about the error. Once thrown, an Exception is handled by the application or by the default exception handler.

[*Note:* For a description of the exception handling model, see Partition I of the CLI Specification.]

[*Note:* If an application handles exceptions that occur during the execution of a block of application code, the code is required to be placed within a **try** statement. Application code within a **try** statement is a *try block*. Application code that handles Exceptions thrown by a try block is placed within a **catch** statement, and is called a *catch block*. Zero or more catch blocks are associated with a try block, and each catch block includes a type filter that determines the types of Exceptions it handles.

When an Exception occurs in a try block, the system searches the associated catch blocks in the order they appear in application code, until it locates a catch block that handles the Exception. A catch block handles an exception of type *T*, if the type filter of the catch block specifies *T* or any type that *T* derives from. The system stops searching after it finds the first catch block that handles the Exception. For this reason, in application code, a catch block that handles a type must be specified before a catch block that handles its base types, as demonstrated in the example that follows this section. A catch block that handles **System.Exception** is specified last.

If the catch blocks associated with the current try block do not handle the Exception, and the current try block is nested within other try blocks in the current call, the catch blocks associated with the next enclosing try block are searched. If no catch block for the Exception is found, the system searches previous nesting levels in the current call. If no catch block for the Exception is found in the current call, the Exception is passed up the call stack, and the previous stack frame is searched for a catch block that handles the Exception. The search of the call stack continues until the Exception is handled or there are no more frames in the call stack. If the top of the call stack is reached without finding a catch block that handles the Exception, the default exception handler handles it and the application terminates.]
System.Exception types support the following features:

- Human-readable text that describes the error. [*Note:* See **System.Exception.Message** property.]

- The state of the call stack when the Exception was thrown. [*Note:* See the **System.Exception.StackTrace** property.]

- When there is a causal relationship between two or more Exceptions, this information is maintained via the **System.Exception.InnerException** property.

The Base Class Library provides two types that inherit directly from **System.Exception**:

- **System.ApplicationException**

- **System.SystemException**

[*Note:* Most user-defined exceptions derive from **System.ApplicationException**. For more information, see **System.ApplicationException** and **System.SystemException**.]

Example

The following example demonstrates a catch block that is defined to handle **System.ArithmeticException** errors. This catch block also catches **System.DivideByZeroException** errors because **System.DivideByZeroException** derives from **System.ArithmeticException**, and there is no catch block explicitly defined for **System.DivideByZeroException** errors.

[C#]

```
using System;
class ExceptionTestClass {
 public static void Main() {
 int x = 0;
 try {
 int y = 100/x;
 }
 catch (ArithmeticException e) {
 Console.WriteLine("ArithmeticException Handler: {0}",
e.ToString());
 }
 catch (Exception e) {
 Console.WriteLine("Generic Exception Handler: {0}",
e.ToString());
 }
 }
}
```

The output is

```
ArithmeticException Handler: System.DivideByZeroException:
Attempted to divide by zero.
   at ExceptionTestClass.Main()
```

Exception() Constructor

```
[ILASM]
public rtspecialname specialname instance void .ctor()

[C#]
public Exception()
```

Summary

Constructs and initializes a new instance of the **System.Exception** class.

Description

This constructor initializes the **System.Exception.Message** property of the new instance to a system-supplied message that describes the error and takes into account the current system culture. The **System.Exception.InnerException** property is initialized to **null** and the **System.Exception.StackTrace** property is initialized to **System.String.Empty**.

Exception(System.String) Constructor

```
[ILASM]
public rtspecialname specialname instance void .ctor(string
message)

[C#]
public Exception(string message)
```

Summary

Constructs a new instance of the **System.Exception** class.

Parameters

Parameter	Description
message	A **System.String** that describes the error. The content of *message* is intended to be understood by humans. The caller of this constructor is required to ensure that this string has been localized for the current system culture.

Description

This constructor initializes the **System.Exception.Message** property of the new instance using *message*. If *message* is **null**, the **System.Exception.Message** property is initialized to the system-supplied message provided by the constructor that takes no arguments. The **System.Exception.InnerException** property is initialized to **null** and the **System.Exception.StackTrace** property is initialized to **System.String.Empty**.

Exception(System.String, System.Exception) Constructor

```
[ILASM]
public rtspecialname specialname instance void .ctor(string
message, class System.Exception innerException)

[C#]
public Exception(string message, Exception innerException)
```

Summary

Constructs a new instance of the **System.Exception** class.

Parameters

Parameter	Description
message	A **System.String** that describes the error. The content of *message* is intended to be understood by humans. The caller of this constructor is required to ensure that this string has been localized for the current system culture.
innerException	An instance of **System.Exception** that is the cause of the current exception. If *innerException* is non-null, then the current exception was raised in a catch block handling *innerException*.

Description

This constructor initializes the **System.Exception.Message** property of the new instance using *message*, and the **System.Exception.InnerException** property using *innerException*. If *message* is **null**, the **System.Exception.Message** property is initialized to the system-supplied message provided by the constructor that takes no arguments.

The **System.Exception.StackTrace** property is initialized to **System.String.Empty**.

Exception.GetBaseException() Method

```
[ILASM]
.method public hidebysig virtual class System.Exception
GetBaseException()

[C#]
public virtual Exception GetBaseException()
```

Summary

Returns the **System.Exception** that is the root cause of one or more subsequent Exceptions.

Return Value

Returns the first Exception thrown in a chain of Exceptions. If the **System.Exception.InnerException** property of the current Exception is **null**, returns the current Exception.

Description

[*Note:* A chain of Exceptions consists of a set of Exceptions such that each Exception in the chain was thrown as a direct result of the Exception referenced in its **System.Exception.InnerException** property. For a given chain, there can be exactly one Exception that is the root cause of all other Exceptions in the chain. This Exception is called the *base exception* and its **System.Exception.InnerException** property always contains a null reference.]

Behaviors

For all Exceptions in a chain of Exceptions, the **System.Exception.GetBaseException** method is required to return the same object (the *base exception*).

How and When to Override

The **System.Exception.GetBaseException** method is overridden in classes that require control over the exception content or format.

Usage

Use the **System.Exception.GetBaseException** method when you want to find the root cause of an Exception but do not need information about Exceptions that may have occurred between the current Exception and the first Exception.

Example

The following example shows an implementation of the **System.Exception.GetBaseException** method.

[C#]

```
public virtual Exception GetBaseException() {
 Exception inner = InnerException;
 Exception back = this;
 while (inner != null) {
 back = inner;
 inner = inner.InnerException;
 }
 return back;
}
```

Exception.ToString() Method

[ILASM]
```
.method public hidebysig virtual string ToString()
```

[C#]
```
public override string ToString()
```

Summary

Creates and returns a **System.String** representation of the current Exception.

Return Value

A **System.String** representation of the current Exception.

Behaviors

System.Exception.ToString returns a representation of the current Exception that is intended to be understood by humans. Where the Exception contains culture-sensitive data, the string representation returned by **System.Exception.ToString** is required to take into account the current system culture. [*Note:* Although there are no exact requirements for the format of the returned string, it should as much as possible reflect the value of the object as perceived by the user.]

[*Note:* This method overrides **System.Object.ToString**.]

Default

The **System.Exception.ToString** implementation obtains the fully qualified name of the current Exception, the message, the result of calling **System.Exception.ToString** on the inner exception, and the result of calling **System.Environment.StackTrace**. If any of these members is **null** or equal to **System.String.Empty**, its value is not included in the returned string.

How and When to Override

It is recommended, but not required, that **System.Exception.ToString** be overridden to return information about members declared in the derived class. For example, the **System.ArgumentException** class overrides **System.Exception.ToString** so that it returns the value of the **System.ArgumentException.ParamName** property, if that value is not **null**.

Usage

Use the **System.Exception.ToString** method to obtain a string representation of an Exception.

Example

The following example causes an Exception and displays the result of calling **System.Exception.ToString** on that Exception.

[C#]

```
using System;
public class MyClass {}
public class ArgExceptionExample {
 public static void Main() {
 MyClass my = new MyClass();
 string s = "sometext";
 try {
 int i = s.CompareTo(my);
 }
 catch (Exception e) {
 Console.WriteLine("Error: {0}",e.ToString());
 }
 }
}
```

The output is

```
Error: System.ArgumentException: Object must be of type
String.
   at System.String.CompareTo(Object value)
   at ArgExceptionExample.Main()
```

Exception.InnerException Property

[ILASM]

```
.property class System.Exception InnerException { public
hidebysig specialname instance class System.Exception
get_InnerException() }
```

[C#]

```
public Exception InnerException { get; }
```

Summary

Gets the **System.Exception** instance that caused the current Exception.

Property Value

An instance of **System.Exception** that describes the error that caused the current Exception.

Description

This property is read-only.

[*Note:* When an Exception *X* is thrown as a direct result of a previous exception *Y*, the **System.Exception.InnerException** property of *X* should contain a reference to *Y*.]

The **System.Exception.InnerException** property returns the same value as was passed into the constructor, or **null** if the inner exception value was not supplied to the constructor.

[*Note:* Using the **System.Exception.InnerException** property, you can obtain the set of Exceptions that led to the current Exception. **System.Exception.GetBaseException** includes an example that demonstrates this procedure.]

Example

The following example demonstrates throwing and catching an Exception that references an inner Exception.

[C#]

```
using System;
public class MyAppException:ApplicationException {
 public MyAppException (String message): base (message) {}
 public MyAppException (String message, Exception inner):
base(message,inner) {}
}
public class ExceptExample {
 public void ThrowInner () {
 throw new MyAppException("ExceptExample inner exception");
 }
 public void CatchInner() {
 try {
 this.ThrowInner();
 }
 catch (Exception e) {
 throw new MyAppException("Error caused by trying
ThrowInner.",e);
 }
 }
}
public class Test {
 public static void Main() {
 ExceptExample testInstance = new ExceptExample();
 try {
 testInstance.CatchInner();
 }
 catch(Exception e) {
 Console.WriteLine ("In Main catch block. Caught: {0}",
e.Message);
 Console.WriteLine ("Inner Exception is
{0}",e.InnerException);
 }
}
}
```

The output is

```
In Main catch block. Caught: Error caused by trying
ThrowInner.
Inner Exception is MyAppException: ExceptExample inner
exception
   at ExceptExample.ThrowInner()
   at ExceptExample.CatchInner()
```

Exception.Message Property

[ILASM]

```
.property string Message { public hidebysig virtual
specialname string get_Message() }
```

[C#]

```
public virtual string Message { get; }
```

Summary

Gets a **System.String** containing a message that describes the current Exception.

Property Value

A **System.String** that contains a detailed description of the error, or **System.String.Empty**. This value is intended to be understood by humans.

Description

[*Note:* The text of **System.Exception.Message** should completely describe the error and should, when possible, explain how to correct it.

The value of the **System.Exception.Message** property is included in the information returned by **System.Exception.ToString**.] This property is read-only.

Behaviors

The **System.Exception.Message** property is set only when creating an Exception instance.

If no message was supplied to the constructor for the current instance, the system supplies a default message that is formatted using the current system culture.

How and When to Override

The **System.Exception.Message** property is overridden in classes that require control over message content or format.

Usage

Application code typically accesses this property when there is a need to display information about an exception that has been caught.

Exception.StackTrace Property

```
[ILASM]
.property string StackTrace { public hidebysig virtual
specialname string get_StackTrace() }

[C#]
public virtual string StackTrace { get; }
```

Summary

Gets a **System.String** representation of the frames on the call stack at the time the current Exception was thrown.

Property Value

A **System.String** that describes the contents of the call stack.

Description

[*Note:* **System.Exception.StackTrace** may not report as many method calls as expected, due to code transformations, such as inlining, that occur during optimization.]

This property is read-only.

Behaviors

The format of the information returned by this property is required to be identical to the format of the information returned by **System.Environment.StackTrace**.

How and When to Override

The **System.Exception.StackTrace** property is overridden in classes that require control over the stack trace content or format.

Usage

Use the **System.Exception.StackTrace** property to obtain a string representation of the contents of the call stack at the time the exception was thrown.