Exception Handling in VB.NET

Kavita K. Bharti
Assistant Professor
Department of Computer
Durga Mahavidyalaya, Raipur

What is an Exception?

An **exception** is an unwanted error that occurs during the execution of a program and can be a system exception or application exception. Exceptions are nothing but some abnormal and typically an event or condition that arises during the execution, which may interrupt the normal flow of the program.

An exception can occur due to different reasons, including the following:

- A user has entered incorrect data or performs a division operator, such as an attempt to divide by zero.
- A connection has been lost in the middle of communication, or system memory has run out.

Exception Handling

When an error occurred during the execution of a program, the exception provides a way to transfer control from one part of the program to another using **exception handling** to handle the error. VB.NET exception has four built-in keywords such as **Try, Catch, Finally**, and **Throw** to handle and move controls from one part of the program to another.

Keyword	Description
Try	A try block is used to monitor a particular exception that may throw an exception within the application.
Catch	It is a block of code that catches an exception with an exception handler at the place in a program where the problem arises.
Finally	It is used to execute a set of statements in a program, whether an exception has occurred.
Throw	As the name suggests, a throw handler is used to throw an exception after the occurrence of a problem.

Exception Classes in VB.NET

In VB.net, there are various types of exceptions represented by classes. And these exception classes originate from their parent's class 'System.Exception'.

The following are the two exception classes used primarily in VB.NET.

1. **System.SystemException**
2. **System.ApplicationException**

System.SystemException: It is a base class that includes all predefined exception classes, and some system-generated exception classes that have been generated during a run time such as **DivideByZeroException, IndexOutOfRangeException, StackOverflowExpression**, and so on.

System.ApplicationException: It is an exception class that throws an exception defined within the application by the programmer or developer. Furthermore, we can say that it is a user-defined exception that inherits from **System.ApplicationException class**.

Syntax of exception handler block

```
Try
 ' code or statement to be executed
  [ Exit Try block]
        ' catch statement followed by Try block
        Catch [ Exception name] As [ Exception Type]
        [Catch1 Statements] Catch [Exception name] As [Exception Type]
        [ Exit Try ]
         [ Finally
         [ Finally Statements ] ]
End Try
```

In the above syntax, the Try/Catch block is always surrounded by a code that can throw an exception. And the code is known as a protected code. Furthermore, we can also use multiple catch statements to catch various types of exceptions in a program, as shown in the syntax.

Example to Exception Handle

Let's create a program to handle an exception using the Try, Catch, and Finally keywords for Dividing a number by zero in VB.NET programming.

TryException.vb

```
Module mod1
Sub ExZero (ByVal a As Integer, ByVal b As Integer)
   Dim res As Integer
   Try
      res = a \ b
      ' Catch block followed by Try block
   Catch ex As DivideByZeroException
      Console.WriteLine(" These exceptions were found in the program {0}", ex)
      ' Finally block will be executed whether there is an exception or not.
   Finally
      Console.WriteLine(" Division result is {0}", res)
   End Try
End Sub
Sub Main()
   ExZero(9, 0) ' pass the parameters value
   Console.WriteLine(" Press any key to exit...")
   Console.ReadKey()
End Sub
End Module
```

When we execute this code Exception will be generated.

Creating User-Defined Exceptions

It allows us to create custom exceptions derived from the **ApplicationException** class.
Let's create a program to understand the uses of User-Defined Exceptions in VB.NET Exception Handling.

Usr_Exp .vb

```
Module Usr_Exp
  Public Class StudentIsZeroException : Inherits Exception
     Public Sub New(ByVal stdetails As String)
       MyBase.New(stdetails)
    End Sub
  End Class
  Public Class StudentManagement
    Dim stud As Integer = 0
    Sub ShowDetail()
      If (stud = 0) Then
        Throw (New StudentIsZeroException(" Student roll no 'zero' does not exist"))
      Else
        Console.WriteLine("  Student is {0}", stud)
      End If
    End Sub
  End Class
  Sub Main()
    Dim stdmg As StudentManagement = New StudentManagement()
    Try
      stdmg.ShowDetail()
    Catch ex As StudentIsZeroException
      Console.WriteLine(" StudentIsZeroException {0}", ex.message)
    End Try
    Console.ReadKey()
  End Sub
End Module
```

Output:

Student roll No ‘zero’ does not exist

Using Try-Catch Statement

We can also create a program using the Try-Catch statement in VB.NET to handle the exceptions.

Try_catch.vb

```
Imports System
Module Try_catch
   Sub Main(ByVal args As String())
      Dim strName As String = Nothing
      Try
         If strName.Length > 0 Then ' it thows and exception
            Console.WriteLine(" Name of String is {0}", strName)
         End If
      Catch ex As Exception  ' it cacthes an exception
         Console.WriteLine(" Catch exception in a proram {0}", ex.Message)
      End Try
      Console.WriteLine(" Press any key to exit...")
      Console.ReadKey()
   End Sub
End Module
```