.NET Framework 4.7 and C# 8.0

Lesson 07 : Exception Handling in C#

Capgemini

Lesson Objectives

- In this lesson we will cover the following :
  - Understand the need for Exception Handling.
  - Learn exception handling in C#. Use try, catch and finally blocks.
  - Understand how to create user-defined exceptions.

What is an Exception?

- Definition: An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions during the execution of a program.
- Exceptions are notifications that some error has occurred in the program.
- When an exception occurs you can ignore the exception or you can write code to deal with the exception, this is known as exception handling.
- In this lesson you will learn the concepts of exceptions and exception handling.

Exception Handling in .NET

- While executing a program if a run-time error occurs, an exception is generated; this is usually referred to as an exception being thrown.
- An exception is an object that contains information about the runtime error which has occurred.
- We can use various techniques to act on an exception.
- In general, this is known as exception handling, and it involves writing code that will execute when an exception is thrown.

Overview – Exception Handling Model

- Programming with structured exception handling involves the use of four interrelated entities:
  - Class type that represents details of the exception occurred.
  - A member to throw an instance of the exception class to the caller.
  - A block of code on the caller's side that invokes the exception-prone member.
  - A block of code on the caller's side that processes (or catches) the exception.

Abnormalities Occurring

- What happens if the file cannot be opened?
  - What happens if you cannot determine the length of the file?
  - What happens if enough memory cannot be allocated?
  - What happens if the read fails?
  - What happens if the file cannot be closed?

```
readFile
{
        open the file;
        find its size;
        allocate memory;
        read file into memory;
        close the file;
}
```

Overview

- In C#, exceptions are represented by classes.
  - All exception classes must be derived from the built-in exception class Exception.
    - Exception is part of the System namespace.
  - From Exception, are derived classes that support two general exception categories defined in C#:
    - SystemException – Generated by C# runtime system.
    - ApplicationException – Generated by Application programs.

Overview

- C# exception handling is managed via four keywords:
  - try, catch, throw, and finally.
  - Try block:
    - Contains program statements you wish to monitor for exceptions.
    - If an exception occurs within the try block, it is thrown.
  - Catch block:
    - Your code catches this exception using catch and handles it in some rational manner.
  - Finally block:
    - Any code that you must execute after you exit a try block is put in a finally block.

Overview (contd..)

- An exception generates when your application encounters an exceptional circumstance.
  - Division by zero or low memory warning.
- Flow of control immediately jumps to an associated exception handler, if one is present.
  - If no handler is present, program execution stops with an error message.
- Actions that result in an exception are executed with the try keyword.

Code Snippet

```
try {
    // Code to try here.
}
catch (System.Exception ex) {
    // Code to handle exception here.
}
finally {
    // Code to execute after try (and possibly catch) here.
}
```

Demo

- Demo on Exception Handling

Overview

- Associate more than one catch statement with a try.
- Each catch must catch a different type of exception.
- If you wish to use an Exception class in multiple catch statements, it should be the last catch statement.

Commonly Used Exceptions

- ArrayTypeMismatchException:
  - Type of value stored is incompatible with the array type.
- DivideByZeroException:
  - Division by zero attempted.
- IndexOutOfRangeException:
  - Array index is out of bounds.
- InvalidCastException:
  - A runtime cast is invalid.
- NullReferenceException:
  - Attempt to operate on a null reference (reference that does not refer to an object).

InnerException Property

- It is property of Exception class
- When there are series of exceptions, the most current exception can obtain the prior exception in the InnerException property
- Use the InnerException property to obtain the set of exceptions that led to the current exception.
- You can create a new exception that catches an earlier exception.
- The code that handles the second exception can make use of the additional information from the earlier exception to handle the error more appropriately.

User Defined Exceptions

- Although C#’s built-in exceptions handle most common errors, C#’s exception handling mechanism is not limited to them.
- You can use custom exceptions to handle errors in your own code.
- As a general rule, exceptions you define should be derived from ApplicationException as this is the hierarchy reserved for application-related exceptions.

User Defined Exceptions

```
class MyException : ApplicationException
{    public MyException(string str):base(str)
     {          Console.WriteLine("User defined exception");          }
}
class MyClient
{
     public static void Main()
     {
               try
               {
               throw new MyException("Some error has happened");
               }
               catch(MyException e)
               {
               Console.WriteLine("Exception caught here" + e.ToString());
               }
               Console.WriteLine("LAST STATEMENT");
     }
}
```

Demo

- Demo on User Defined Exceptions

Summary

- In this module you learned:
  - Need for Exception Handling
  - Exception Handling using try, catch and finally block.
  - Creating a User Defined Exception by inheriting it from a General Exception class.

Review Question

- Why do you need exception Handling?
- Can I have multiple catch blocks written for one try block?
- What is the use of finally block?
- How do you create user defined exception?
- What should be placed first, General Exception or specific Exception?