Exception Handling

Agenda

- Exception – Keywords
- Handling exceptions
- Exception classes in C#
- User-defined exceptions
- Best practices in exception handling

Exception - Keywords

- An exception is an undesired problem in the software program that arises during run-time. In C#, the exception is handled by the following keywords:
  - **try** : the software block where the actual code is written.
  - **catch**: A program can have multiple catch blocks and are handled in these catch blocks based on the exception that is raised in the try block of the code.
  - **finally**: In this block all the resources are released. This blocks runs whether or not an exception is thrown in the try block.
  - **throw**: Whenever there is a problem in the code an exception is thrown and is done through the throw keyword.

```
try {
  // statements causing exception
} catch( ExceptionName e1 ) {
  // error handling code
} catch( ExceptionName e2 ) {
  // error handling code
}
finally {
  // statements to be executed
}
```

Handling exceptions

- The exception handling is a strategic design decision and have to be provisioned for, right from Architectural definition of a software program.
- .NET framework provides structured way of handling the errors.
- The keywords try, catch, finally blocks provides ways to separate the core program from error handling statements.
- Exception filters can be used to multiple exceptions using predefined exception classes from .NET framework
- When making calls across methods between two different components the **throw** keyword can be used to pass the exceptions from the callee to the caller methods.

Let’s demonstrate the exception concept with an example program.

Exception classes in C#

- The .NET framework provides following predefined exception classes derived from System.SystemException class. .

Exception class	Description
System.IO.IOException	Handles I/O errors
System.IndexOutOfRangeException	When a method refers an array which is out of index, error raised would be handled using this exception
System.ArrayTypeMismatchException	Handles errors generated when type is mismatched with the array type
System.NullReferenceException	Errors generated from referencing a null object is handled
System.DivideByZeroException	Errors generated from dividing a dividend with zero is handled
System.InvalidCastException	Handles errors generated while typecasting.
System.OutOfMemoryException	Handles errors arising out of insufficient memory.
System.StackOverflowException	Handles errors generated from stack overflow.

User-defined exceptions

- It is also possible to have user-defined exceptions in C#. User-defined classes have to be derived from System.Exception class.

Let’s demonstrate the above concept with an example program.

Best practices in exception handling

- The classes are to be designed in such a way to avoid exceptions.
- It is better to use the predefined .NET exceptions.
- It is always a good practice to catch specific exception using Exception filters rather than catching an generic exception of type Exception.
- Inclusion of a localized string message with every exception.
- Usage of builder exception methods

Let’s demonstrate the above concepts with an example program.

Summary

In this tutorial we have covered exceptions and how to handle them. We also learnt how to create user-defined exceptions, various built-in exceptions classes provided by .NET framework and some best practices in exception handling.

Thank You