Microsoft

LESSON 1.3

10754 Microsoft .NET Fundamentals

Understand Exception Handling in the .NET Framework

Lesson Overview

How does the Microsoft® .NET Framework use structured exception handling?

In this lesson, you will explore:

- Basic types of programming errors
- Exceptions and how to handle them in code

Guiding Questions

1. What are some common errors?
2. What is an exception?
3. What are the common exceptions?
4. How can developers write code to handle exceptions gracefully?

Anticipatory Set

Consider the code below. What will happen when it is executed?

```
int numerator = 0;
int denominator = 0;
int answer = 0;
numerator = 3;
answer = numerator / denominator;
denominator = 4;
Console.WriteLine(answer);
```

Types of Errors

- **Syntax errors** occur when code does not meet the rules (or "syntax") of the programming language in use.
  - For example, using incorrect case (such as 'If' in C# code) is a syntax error.
  - Syntax errors occur at compile time. An application cannot compile—and therefore cannot run—if the code contains syntax errors.
- **Logic errors** occur when code executes but does not behave in the intended manner. Consider the following `if` statement:

```
if (x < 0 && x > 10)
```

  This code follows C# syntax, and it will not crash the application. However, it is not likely to be what the developer intended.
  - Logic errors are often simply called bugs.
- **Exceptions** occur when something unexpected happens that disrupts the flow of the application.

Exceptions

- An **exception** is an event that is raised when a method encounters a condition that prevents it from executing.
- Often, this is due to invalid data.
- Exceptions that are not handled will cause the application to terminate.

Common Types of Exceptions

There are many different types of exceptions. Commonly encountered exceptions include:

- A *DivideByZeroException*
  - Occurs when there is an attempt to divide a value by zero, as shown in the Anticipatory Set.
- A *FileNotFoundException*
  - Occurs when an attempt to access a file that does not exist on disk fails. This would happen if the code refers to an incorrect file name or path, or the file does not exist.
- An *IndexOutOfBoundsException*
  - Occurs when an attempt is made to read beyond the bounds of a collection. This would happen if code references index number 8 in an array of only three items.

Handling Exceptions

- Structured exception handling allows the developer to address potential exceptions so that the application doesn't terminate, or at least it allows for a graceful termination of the application.
- Exception handling uses the `try`, `catch`, and `finally` keywords (in Visual Basic®: `Try`, `Catch`, and `Finally`) to attempt actions that may not succeed, to handle failures, and to clean up resources afterwards.

Note: Programmers often use the term "throw" (rather than "raise") when talking about exceptions.

For example, the code from the Anticipatory Set in the presentation throws a *DivideByZeroException*.

`try`

- The `try` (or `Try`) keyword is used to indicate that a block of code should be monitored for exceptions.
  - You can think of it as giving a warning to the application that there may be trouble ahead—so watch out!

`catch`

- The `catch` (or `Catch`) keyword is used to designate a block of code used to handle an exception if one is thrown.
  - You can use multiple `catch` blocks to handle different types of exceptions.

Example of a Method with `try/catch` Blocks

```
int SafeDivision(int x, int y)
{
 try
 {
     return (x / y);
 }
 catch (System.DivideByZeroException dbz)
 {
    System.Console.WriteLine("Division by zero!");
    return 0;
 }
}
```

`finally`

- The code in a `finally` (or `Finally`) block always executes last, just before the error-handling block loses scope, regardless of whether the code in the `catch` blocks has executed.
- `finally` blocks are not mandatory in structured exception handling, but they provide a great place for cleanup code (such as code for storing important data, closing files, etc).

Lesson Review

- List three basic types of errors.
- What are exceptions?
- What keywords are used to handle exceptions in applications?