**在查询表达式中处理异常（C# 编程指南）**

可以在查询表达式的上下文中调用任何方法。但是，建议您避免在查询表达式中调用任何可能产生副作用（例如，修改数据源的内容或引发异常）的方法。此示例演示在查询表达式中调用方法时如何避免引发异常，同时又不违反 .NET Framework 的有关异常处理的一般性准则。这些准则说明：如果您了解为什么在给定上下文中会引发特定异常，那么就可以捕捉该异常。有关更多信息，请参见[异常的最佳做法](https://msdn.microsoft.com/zh-cn/library/seyhszts.aspx)。

最后一个示例演示如何处理那些必须在查询执行过程中引发异常的情况。

下面的示例演示如何将异常处理代码移至查询表达式外部。仅当该方法不依赖于查询的任何本地变量时，才能这样做。

class ExceptionsOutsideQuery

{

static void Main()

{

// DO THIS with a datasource that might

// throw an exception. It is easier to deal with

// outside of the query expression.

IEnumerable<int> dataSource;

try

{

dataSource = GetData();

}

catch (InvalidOperationException)

{

// Handle (or don't handle) the exception

// in the way that is appropriate for your application.

Console.WriteLine("Invalid operation");

goto Exit;

}

// If we get here, it is safe to proceed.

var query = from i in dataSource

select i \* i;

foreach (var i in query)

Console.WriteLine(i.ToString());

//Keep the console window open in debug mode

Exit:

Console.WriteLine("Press any key to exit");

Console.ReadKey();

}

// A data source that is very likely to throw an exception!

static IEnumerable<int> GetData()

{

throw new InvalidOperationException();

}

}

在某些情况下，对在查询内引发的异常的最佳响应可能是立即停止执行查询。下面的示例演示如何处理可能从查询正文内部引发的异常。假定 SomeMethodThatMightThrow 可能导致要求停止执行查询的异常。

请注意，**try** 块将 **foreach** 循环而不是查询本身封闭起来。这是因为 **foreach** 循环是实际执行查询的场所。有关更多信息，请参见 [Introduction to LINQ Queries (C#)](https://msdn.microsoft.com/zh-cn/library/bb397906.aspx)。

class QueryThatThrows

{

static void Main()

{

// Data source.

string[] files = { "fileA.txt", "fileB.txt", "fileC.txt" };

// Demonstration query that throws.

var exceptionDemoQuery =

from file in files

let n = SomeMethodThatMightThrow(file)

select n;

// Runtime exceptions are thrown when query is executed.

// Therefore they must be handled in the foreach loop.

try

{

foreach (var item in exceptionDemoQuery)

{

Console.WriteLine("Processing {0}", item);

}

}

// Catch whatever exception you expect to raise

// and/or do any necessary cleanup in a finally block

catch (InvalidOperationException e)

{

Console.WriteLine(e.Message);

}

//Keep the console window open in debug mode

Console.WriteLine("Press any key to exit");

Console.ReadKey();

}

// Not very useful as a general purpose method.

static string SomeMethodThatMightThrow(string s)

{

if (s[4] == 'C')

throw new InvalidOperationException();

return @"C:\newFolder\" + s;

}

}

/\* Output:

Processing C:\newFolder\fileA.txt

Processing C:\newFolder\fileB.txt

Operation is not valid due to the current state of the object.

\*/