《现代软件开发——使用.NET与C#》系列第五讲

C# 中的异常处理

俞晖
MSDN讲师
dolphin@vip.163.com

《现代软件开发——使用.NET与C#》系列

- 回顾上节课的内容（*类的设计*）
- 本节课讲述在**C#**中的异常处理
- **LEVEL 200**

Date	Topic
2005年7月21日	*C# 中的异常处理*
2005年8月1日	*WinForms*
	ADO.NET
.	.
.	.

日程

- 异常概述（Exception）
- 捕获异常及处理
- 跟踪（Trace）

引子

- 用户输入就是一个很好的例子
- 问题:
  - 我们需要用户输入一个integer，用户真的就输入了一个integer吗？
  - 如果这个integer太大了或者太小了，怎么办？

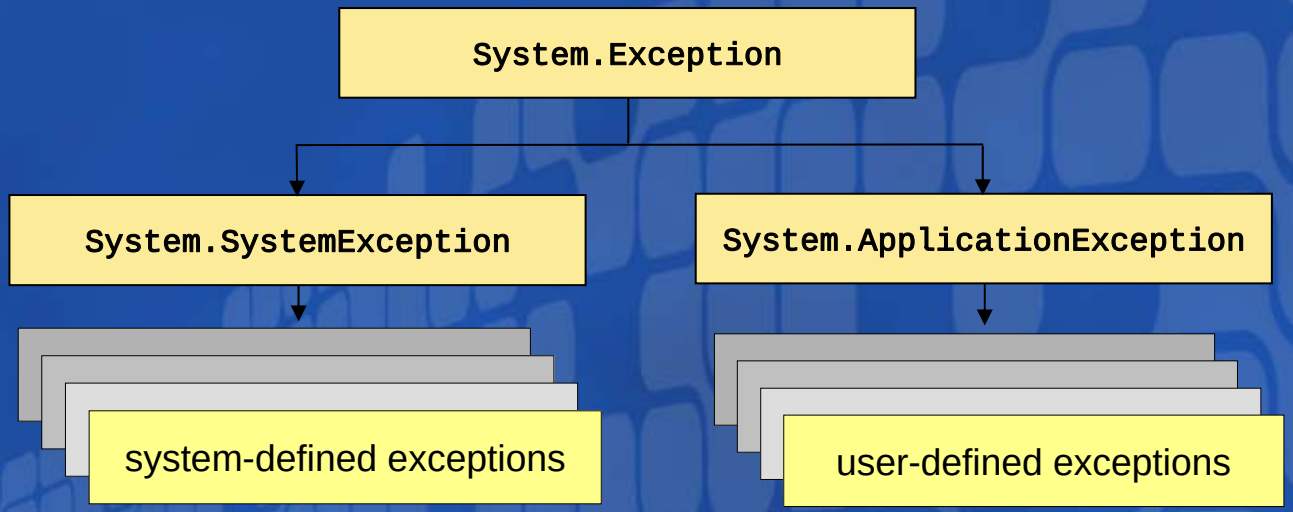
```
string  s;
int     i;

s = << user's input >>;
i = int.Parse(s);
```

异常（Exception）

- 异常是当程序发生错误时产生的一种信号
- 在.NET中，广泛使用，为什么？
- **Examples:**
  - divide-by-zero
  - arithmetic overflow
  - array access out of bounds
  - null object reference
  - file not found

异常类型

- 每种异常类型都是一个类
- 两种大分类
  - *System.SystemException*
  - *System.ApplicationException*



.NET中异常处理方式

- .NET中异常处理方式
  - 异常被对象所表现而不是错误代码
  - 异常的产生是通过*throwing一个该异常的对象实现的*
  - 异常的捕获是通过*catch*该异常的对象
  - 命名上可以读出是哪类异常：
    - *ArithmeticException*, *DivideByZeroException*, etc.

捕获异常try-catch

- 当代码段有可能发生异常的时候，我们应该把该代码段放置在try中
- 捕获到异常后的处理方法放置到catch中。

```
try
{
    s = << user's input >>;
    i = int.Parse(s);    // if this fails, .NET throws exception…
}
catch
{
    MessageBox.Show("Invalid input, please try again.");
    return;    // return now and give user another chance…
}
```

异常处理

demo

- 用户输入异常

更好的解决方案

- 为每个可能的Exception定制解决方法

```
try
{
   s = << user's input >>;
   i = int.Parse(s);   // if this fails, .NET throws exception…
}
catch(FormatException)
{
   MessageBox.Show("Please enter a numeric value.");
   return;
}
catch(OverflowException)
{
   MessageBox.Show("Value is too large/small (valid range " + int.MinValue +
                   " ... " + int.MaxValue + ").");
   return;
}
catch(Exception ex)
{  // else something we didn't predict has happened…
   string  msg;
   msg = String.Format("Invalid input.\n\n[Error={0}]", ex.Message);
   MessageBox.Show(msg);
   return;
}
```

exception

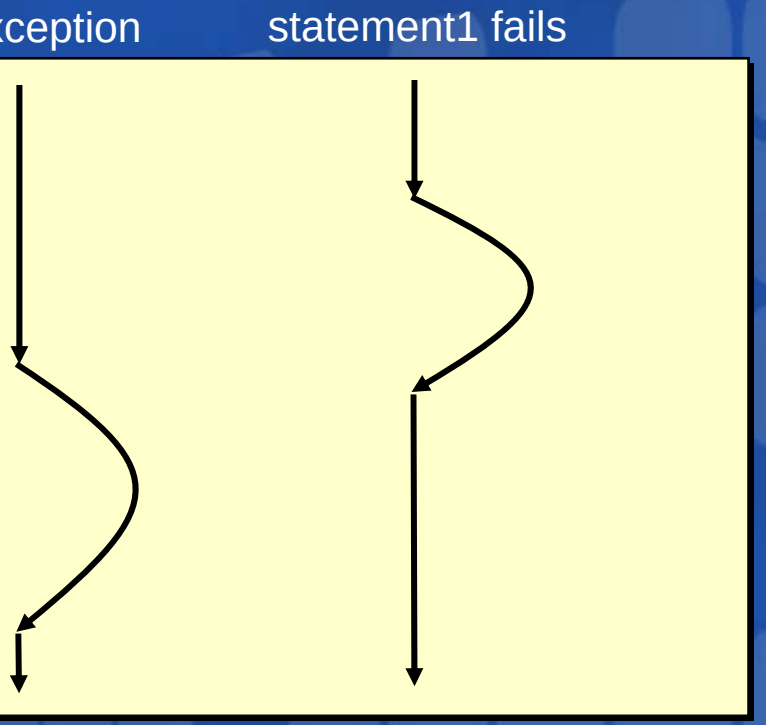
异常处理的系统流程

- 当程序产生一个异常的时候，它会自动抛出异常，此时.NET进入“异常处理模式”
  - .NET 查找后面是否存在catch子句
  - If (catch block is found)
    .NET executes catch & “exception mode” is over
    else *// no catch block to handle exception…*
    .NET terminates execution
- 暗示
  - 如果你不想让程序被错误所终止，你要在适当的地方使用try-catch
  - 如果您想让异常处理继续，你要在catch子句中写出一些具体的方法，空的catch段相当于给异常放行。

try-catch的工作方式

- 控制流的两条走向：
  - 无异常： 跳过**catch**段直接到**setp4**
  - 异常：立刻直接进入**catch**段

```
try
{
    step1;
    step2;
    step3;
}
catch (Exception ex)
{
    catch1;
    catch2;
}
step4;
```



暗示

- 在进行完catch子句后，程序将继续执行
  - ... 除非在catch子句中有 *return*, *throw*, 或者 *exit*

```
try
{
    :
}
catch(Exception
{
    :
    return;
}
:
```

```
try
{
    :
}
catch(Exception
{
    :
    throw ex;
}
:
```

```
try
{
    :
}
catch(Exception ex)
{
    :
    System.Environment.Exit(1);
}
:
```

异常处理

demo

- 异常的处理

实践

- 嵌套的try-catch

```
try
{
   << perform operation(s) >>
}
catch(Exception ex)
{
    try
    {
        << try various recovery strategies >>
    }
    catch
    {
        // none of the strategies worked…
        throw ex;
    }
}
```

异常捕获顺序

- 必须正确排列捕获异常的catch子句
  - 范围小的Exception放在前面的catch子句
  - 即如果Exception之间有继承关系，把之类放在前面的catch子句中，把父类放到后面的catch子句中

```
try
{
   << open database connection >>
   << perform database work >>
}
catch(FileNotFoundException fe)
{
   << deal with any exceptions >>
}
catch(Exception e)
{
 << deal with any exceptions >>
}
```

try-catch-finally

- 使用 try-catch-finally 来确保一些收尾工作
  - finally 段里的语句总在try or catch之后执行

```
try
{
   << open database connection >>
   << perform database work >>
}
catch
{
   << deal with any exceptions >>
}
finally
{
    << close database >>
}
```

抛出异常 Throw Exceptions

- 类中可以抛出异常
  - 不返回错误代码，不输出错误信息
  - 抛出特定的类型

```
public class BankCustomer
{
    .
    .
    .
   public void Withdraw(decimal amt)
   {
      if (amt < 0.0M)
        throw new ArgumentException("Amount cannot be negative.");
      else if (amt > this.m_Balance)   // insufficient funds!
        throw new ApplicationException("Insufficient funds!");
      else
        this.m_Balance -= amt;
   }
```

设计自己的异常

- 创建独特的异常，使它适合于特定的应用程序

```
public class TooManyItemsException : Exception
{
    public TooManyItemsException() : base(@" You add too many items
            to this container.")
    {}
}
```

```
public class TooManyItemsTest
{

    public void GenerateException(int i)
    {
        try
        {
            if(i<2){throw new TooManyItemsException();
        }
    }
}
```

跟踪 Tracing

- *跟踪* == 运行时输出信息
  - 当运行出错时，我们有信息可以参考
  - 是数据记录器
- 思路:
  - 程序中输出要输出的信息
  - 启动时可选：**Enable/disable**

配置跟踪

- 最常用的方法是在.NET config中配置
  Example:
  - 如果应用程序是 *App.exe*, 那么 config 文件就是 *App.exe.config*
  - 在*appSettings*区中添加 (key, value) 值对
  - 放置 .config 文件于.exe的目录下

```
<configuration>

   <appSettings>
      <add key="Tracing" value="true" />
   </appSettings>

</configuration>
```

在VS中创建.config文件

- 通常我们把.config 放置在.EXE同目录下
  - bin\Debug and bin\Release
- 不过，如果你这么做，，Visual Studio 将删除它!
- 所以，通过项目来建立 *app.config*
  - Project menu, Add New Item..., *Application Config File*
  - edit .config file appropriately
  - VS will copy to bin\Debug & rename

跟踪执行


```
using T = System.Diagnostics.Trace;

public class DataAccess
{
  :
  :
  public void Open()
  {
     try {
        T.WriteLineIf(Globals.Trace, “DataAccess.Open():”);
        T.Indent();
        T.WriteLineIf(Globals.Trace, “Connection string=...”);
        ...
     }
     catch(Exception ex) {
        T.WriteLineIf(Globals.Trace, ex.GetType() + “: ” + ex.Message);
        T.WriteLineIf(Globals.Trace, ex.StackTrace);
        ...
     }
     finally {
        T.Unindent();
        T.Flush();
     }
  }
```

demo

- Tracing…

其它参考资源

- **Webcast:**
  - http://www.microsoft.com/china/msdn/events/webcasts/shared/Webcast/MSDNWebCast.aspx
- **Resources:**
  - MSDN Library
    http://www.microsoft.com/china/MSDN/library/default.mspx
  - GotDotNet

    http://www.gotdotnet.com

MSDN flash:

- http://www.microsoft.com/china/newsletter/case/MSDN.asp

感谢大家的参与!

- 非常感谢大家的参与
- 您的反馈对我们来说是非常重要的

Date	Topic
2005年7月7日10:00-11:30	*C#中类的设计*
2005年7月21日	C#中异常处理
2005年8月1日.	C#中WinForm应用


您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts

Q&A

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。