C# 正则表达式 C# 文件的输入与输出

C# 异常处理

异常是在程序执行期间出现的问题。C# 中的异常是对程序运行时出现的特殊情况的一种响应，比如尝试除以零。

异常提供了一种把程序控制权从某个部分转移到另一个部分的方式。C# 异常处理时建立在四个关键词之上的：**try**、**catch**、**finally** 和 **throw**。

- **try**：一个 try 块标识了一个将被激活的特定的异常的代码块。后跟一个或多个 catch 块。
- **catch**：程序通过异常处理程序捕获异常。catch 关键字表示异常的捕获。
- **finally**：finally 块用于执行给定的语句，不管异常是否被抛出都会执行。例如，如果您打开一个文件，不管是否出现异常文件都要被关闭。
- **throw**：当问题出现时，程序抛出一个异常。使用 throw 关键字来完成。

语法

假设一个块将出现异常，一个方法使用 try 和 catch 关键字捕获异常。try/catch 块内的代码为受保护的代码，使用 try/catch 语法如下所示：

```
try
{
   // 引起异常的语句
}
catch( ExceptionName e1 )
{
   // 错误处理代码
}
catch( ExceptionName e2 )
{
   // 错误处理代码
}
catch( ExceptionName eN )
{
   // 错误处理代码
}
finally
{
   // 要执行的语句
}
```

您可以列出多个 catch 语句捕获不同类型的异常，以防 try 块在不同的情况下生成多个异常。

C# 中的异常类

C# 异常是使用类来表示的。C# 中的异常类主要是直接或间接地派生于 **System.Exception** 类。**System.ApplicationException** 和 **System.SystemException** 类是派生于 System.Exception 类的异常类。

System.ApplicationException 类支持由应用程序生成的异常。所以程序员定义的异常都应派生自该类。

System.SystemException 类是所有预定义的系统异常的基类。

下表列出了一些派生自 Sytem.SystemException 类的预定义的异常类：

异常类	描述
System.IO.IOException	处理 I/O 错误。
System.IndexOutOfRangeException	处理当方法指向超出范围的数组索引时生成的错误。
System.ArrayTypeMismatchException	处理当数组类型不匹配时生成的错误。
System.NullReferenceException	处理当依从一个空对象时生成的错误。
System.DivideByZeroException	处理当除以零时生成的错误。
System.InvalidCastException	处理在类型转换期间生成的错误。
System.OutOfMemoryException	处理空闲内存不足生成的错误。
System.StackOverflowException	处理栈溢出生成的错误。

异常处理

C# 以 try 和 catch 块的形式提供了一种结构化的异常处理方案。使用这些块，把核心程序语句与错误处理语句分离开。

这些错误处理块是使用 **try**、**catch** 和 **finally** 关键字实现的。下面是一个当除以零时抛出异常的实例：

实例

```
using System;
namespace ErrorHandlingApplication
{
    class DivNumbers
    {
        int result;
        DivNumbers()
        {
            result = 0;
        }
        public void division(int num1, int num2)
        {
            try
            {
                result = num1 / num2;
            }
            catch (DivideByZeroException e)
            {
                Console.WriteLine("Exception caught: {0}", e);
            }
            finally
            {
                Console.WriteLine("Result: {0}", result);
            }
```

```
        }
        static void Main(string[] args)
        {
            DivNumbers d = new DivNumbers();
            d.division(25, 0);
            Console.ReadKey();
        }
    }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Exception caught: System.DivideByZeroException: Attempted to divide by zero.
at ...
Result: 0
```

创建用户自定义异常

您也可以定义自己的异常。用户自定义的异常类是派生自 **ApplicationException** 类。下面的实例演示了这点：

实例

```
using System;
namespace UserDefinedException
{
   class TestTemperature
   {
      static void Main(string[] args)
      {
         Temperature temp = new Temperature();
         try
         {
            temp.showTemp();
         }
         catch(TempIsZeroException e)
         {
            Console.WriteLine("TempIsZeroException: {0}", e.Message);
         }
         Console.ReadKey();
      }
   }
}
public class TempIsZeroException: ApplicationException
{
   public TempIsZeroException(string message): base(message)
   {
   }
}
public class Temperature
{
   int temperature = 0;
```

```
   public void showTemp()
   {
      if(temperature == 0)
      {
         throw (new TempIsZeroException("Zero Temperature found"));
      }
      else
      {
         Console.WriteLine("Temperature: {0}", temperature);
      }
   }
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
TempIsZeroException: Zero Temperature found
```

抛出对象

如果异常是直接或间接派生自 **System.Exception** 类，您可以抛出一个对象。您可以在 catch 块中使用 throw 语句来抛出当前的对象，如下所示：

```
Catch(Exception e)
{
   ...
   Throw e
}
```

C# 正则表达式 C# 文件的输入与输出