第69章 独自の例外クラスを作る

既存の例外クラスを継承して独自の例外クラスを作ることができます。

この場合、独自の例外クラスのオブジェクトを明示的に投げる必要があります。

```
throw 例外オブジェクト;
```

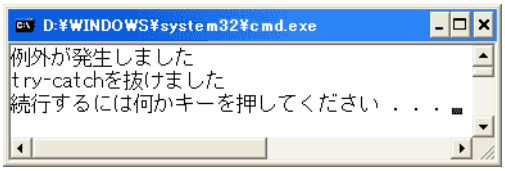
では、サンプルを見てみましょう。

```
// throw01.cs

using System;

class MyException : ApplicationException
{
    public override string Message
    {
        get
        {
            return "例外が発生しました";
        }
    }
}

class throw01
{
    public static void Main()
    {
        try
        {
            throw new MyException();
        }
        catch (MyException me)
        {
            Console.WriteLine(me.Message);
        }
        Console.WriteLine("try-catchを抜けました");
    }
}
```

MyExceptionクラスは、ApplicationExceptionクラスから派生しており、Messageプロパティをオーバーライドしています。

Mainメソッドのtry節ではいきなりMyExceptionクラスのオブジェクトを投げて人為的に例外を発生させています。

この、投げられた例外をcatch節で捕まえています。



実際に、独自の例外クラスを利用する場合は、一旦既存の例外クラスのオブジェクトを捕まえて、そこから独自クラスのオブジェクトを投げるようにします。しかし、この場合例外オブジェクトは、外側のcatch節に向かって投げられる点に注意してください。

では、簡単なサンプルを見てみましょう。

```
// helplink01.cs

using System;

public class MyException : ApplicationException
{
    string help = "http://www.kumei.ne.jp/c_lang/";

    public override string HelpLink
    {
        get
        {
            return help;
        }
```

```
        set
        {
            help = value;
        }
    }
}

class helplink01
{
    public static int MyInput()
    {
        int no;
        Console.Write("整数値を入力--- ");
        string strNo = Console.ReadLine();
        try
        {
            no = Int32.Parse(strNo);
        }
        catch (Exception)
        {
            throw new MyException();
        }
        return no;
    }

    public static void Main()
    {
        int no;

        try
        {
            no = MyInput();
        }
        catch (MyException me)
        {
            Console.WriteLine("入力が不正です¥n" +
                me.HelpLink + "を参照してください");
            no = 0;
        }
        Console.WriteLine("no = {0}", no);

    }
}
```

MyExceptionクラスは、ApplicationExceptionクラスから派生しています。

ApplicationExceptionクラスのHelpLinkプロパティをオーバーライドしています。HelpLinkは、Exceptionクラスからの継承です。

```
public virtual string HelpLink { get; set; }
```

プロパティの値は、例外を説明するURLなどです。

さて、Mainメソッドを有するhelplink01クラスを見てみましょう。

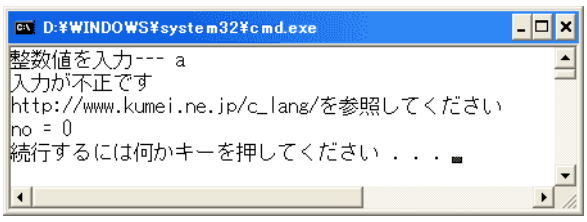
ここには、staticなMyInputメソッドがあります。

このメソッドの中にtry-catch構文があります。

catch節で例外が捕まったら、ここでMyExeceptionのオブジェクトを投げています。この例外オブジェクトは、この外側のtry-catch構造に向かって投げられます。

Mainメソッドでは、MyInputメソッドをtry節の中から呼び出しています。MyInputメソッドで発生すると、ここのcatchブロックで捕まるはずです。

では、入力時にわざとに例外を発生させて、正しく動作するか試してみましょう。



MyInputメソッドで発生した例外が、そこのcatch節で捕まり、そこから投げられた独自の例外オブジェクトが、外側のcatch節で捕まっているのがわかります。

さて、catch節から、例外を投げる場合catch節で捕まえた例外オブジェクトと同じオブジェクトを投げることも可能です。この場合単に

```
throw;
```

とだけ書けばよいのです。

Update 14/Oct/2006 By Y.Kumei