

Globaler Exception-Handler

Notausgang

Auch in den besten Programmen kommt es zu Fehlern. Im schlimmsten Fall führt ein Absturz zu unkontrolliertem Datenverlust. Aber dieser Anwendungs-GAU lässt sich verhindern, wenn ein zentraler Exception-Handler auch unerwartete Fehler abfängt. `dotnetpro` zeigt, wie Sie in Ihre Anwendungen einen Notausgang einbauen.

Trotz diverser Vorsichtsmaßnahmen stürzt ein Programm irgendwann unkontrolliert ab. Man weiß nicht, wo und unter welchen Umständen der Absturz passiert ist, und der Kunde kann ihn nicht reproduzieren.

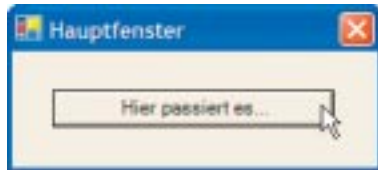


Abbildung 1 Eine Exception auslösen.

Es ist dabei schon eine große Hilfe, wenn die Ausnahme mit den grundlegenden Infos gemeldet und der Anwender gefragt wird, ob er das Programm fortführen oder abbrechen möchte, wie es etwa Abbildung 2 zeigt. Dem Programm selbst fehlt jedoch jegliche Kontrolle darüber. Also kann es nicht entsprechend reagieren oder kontrolliert herunterfahren. An bekannten kritischen Stellen im Pro-

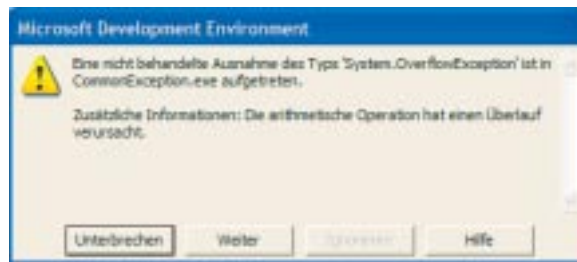


Abbildung 2 Standardmeldung einer Exception.

gramm werden Sie sicherlich mit *Try ... Catch*-Blöcken mögliche Fehler abfangen und entsprechend reagieren. Was aber ist mit Programmteilen, die Sie nicht als kritisch erachten und daher nicht mit besonderen Vorkehrungen für den Fehlerfall ausgestattet haben? Dann verliert das Programm im Fehlerfall die Kontrolle.

Es gibt aber auch die Möglichkeit, einen Exception-Handler zu erstellen, der bei allen unbehandelten Ausnahmen aufgerufen wird. Hier können Sie dann eine Fehlermeldung ausgeben, gegebenenfalls Statusinformationen dabei anzeigen oder protokollieren und das Programm kontrolliert herunterfahren, damit Daten nicht verloren gehen oder inkonsistent werden. Tritt ein Fehler innerhalb eines *Try ... Catch*-Blockes auf, wird natürlich dieser wirksam – der neue Handler ist

wirklich nur für unbehandelte Ausnahmen zuständig.

Sinnvollerweise wird der globale Handler in einem Startmodul untergebracht. Die *Sub Main* enthält dann die Einrichtung des Handlers und den eigentlichen Start des Programms mit dem Hauptfenster der Anwendung. In den Projekteigenschaften muss dieses Modul dann als Startobjekt angegeben werden.

Listing 1 zeigt den Code eines solchen Startmoduls. Zu Beginn wird hier per *AddHandler* die Prozedur *OnThreadException* zur Behandlung der unbehandelten *ThreadExceptions* angemeldet. Danach wird der Hauptdialog *frmMain* instanziiert und das Programm mit dieser Instanz gestartet.

Der hier eingerichtete Handler zeigt den Exception-Text samt Callstack an

Auf einen Blick

Autor

Dipl.-Inform. **Stefan A. Dittrich** ist seit über zehn Jahren als freier Buch- und Software-Autor, Journalist und Unternehmensberater tätig. Sie erreichen ihn unter Mail@SDittrich.de.



dotnetpro.code
A0409Exception



Sprachen VB.NET

Technik Fehlerbehandlung

Voraussetzungen VS.NET

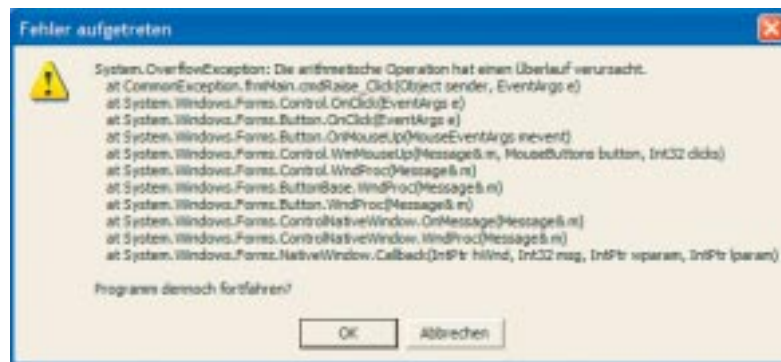


Abbildung 3 Einfache Meldung bei kontrollierter Fehlerbehandlung.

Listing I

Globaler Exception-Handler im Startmodul.

```
*** Startmodul der Anwendung
Module modMain
  *** Objekt für Hauptfenster
  Private fMain As frmMain

  Sub Main()
    *** allgemeinen Exceptionhandler einrichten
    AddHandler System.Windows.Forms.Application.ThreadException, _
      AddressOf OnThreadException

    *** Programm mit Hauptfenster durchstarten
    fMain = New frmMain
    Application.Run(fMain)

  End Sub

  Public Sub OnThreadException(ByVal sender As Object, _
    ByVal t As System.Threading.ThreadExceptionEventArgs)
    *** Allgemeiner Fehlerhandler

    *** Fehlermeldung holen: incl. Callstack
    Dim ErrText As String = t.Exception.ToString
    *** ... oder nur Fehlerbeschreibung
    'ErrText = t.Exception.Message

    *** ggf. auf Konsole ausgeben

    'Console.WriteLine("Exception aufgetreten: " + ErrText)
    *** und/oder im Ausgabefenster der IDE
    'Debug.WriteLine("Exception aufgetreten: " + ErrText)

    *** Fehler melden mit Option Weiter oder Abbrechen
    ErrText = String.Concat(ErrText, vbCrLf, vbCrLf, _
      "Programm dennoch fortfahren?")
    If MsgBox(ErrText, _
      MsgBoxStyle.Exclamation Or MsgBoxStyle.OKCancel, _
      "Fehler aufgetreten") = MsgBoxResult.Cancel Then

      *** Programmende bei "Cancel"
      Try
        *** Fenster schließen
        *** dabei weitere Exceptions ignorieren
        fMain.Close()
      Catch
      End Try

      *** und Schluss!
      Application.Exit()
    End If
  End Sub
End Module
```

und bietet dem Anwender die Möglichkeit, das Programm abzubrechen, wie es Abbildung 3 zeigt. In diesem Fall wird das Hauptfenster kontrolliert geschlossen, wobei Aufräum- und Persistierungsarbeiten erledigt werden können. Hier können natürlich auch zusätzliche Protokollierungen erfolgen, etwa in eine Log-Datei.

Bei Knopfdruck Fehler

Abbildung 1 zeigt ein Beispielprogramm, mit dem die Reaktion des Programms auf eine unbehandelte Ausnahme demonstriert wird. Hierzu wird einfach auf Knopfdruck eine Teilung durch 0 durchgeführt:

```
Private Sub cmdRaise_Click (ByVal sender As _
  System.Object, ByVal e As _
  System.EventArgs) Handles cmdRaise.Click
  *** hier passiert es:
  Dim A As Integer
  Dim B As Integer
  A = 1 / B *** Fehler!
End Sub
```

Ohne den eigenen Handler wird im Ausnahmefall die Meldung aus Abbildung 2 angezeigt, ohne dass das Programm selbst darauf reagieren kann.

Abbildung 3 stellt diese Situation mit dem eigenen Exception-Handler dar. Die hier verwendete MessageBox ist nur eine Minimallösung – besser wäre natürlich ein spezieller Dialog mit größerer Aussagekraft und gegebenenfalls weiteren Absicherungen gegen Folgefehler.

Hier könnten Sie beispielsweise auch Protokollierungen oder den Versand des Fehlers per Mail einbauen, ebenso wie die Option, das Programm neu zu starten. Solche Dialoge kennen Sie vielleicht schon aus Programmen wie MS Office 2002. Klickt der Anwender hier auf *Abbrechen*, wird das Programm kontrolliert beendet – in diesem Beispiel durch das Schließen des Hauptfensters –, wobei dabei auftretende Exceptions – meist Folgefehler – einfach ignoriert werden, gefolgt von *Application.Exit()*.

Fazit

Nichts ist für den Anwender und auch für den Entwickler frustrierender als Abstürze ohne Detailinformationen und weitere Kontrolle des Ablaufs.

Die hier gezeigte Technik des globalen Exception-Handlers macht Ihr Programm nicht nur sicherer und reduziert Datenverluste durch unkontrolliertes Abbrechen, sondern hilft Ihnen auch bei der Fehlersuche. |||||

Anzeige 1/8 quer