Error Handling

Autor & Copyright: Dipl.-Ing. Harald Nahrstedt

Version: 2016 / 2019 / 2021 / 365

Erstellungsdatum: 1.03.2024

Überarbeitung:

Beschreibung:

Laufzeitfehler, die bei Ausführung einer Prozedur auftreten, führen zum Abbruch der Verarbeitung. Weil sich diese Fehler normalerweise nicht unter der Kontrolle des Programmierers befinden und auch die angezeigten Fehlertexte oft wenig Aufschluss über den Sachverhalt wiedergeben, geschweige denn Anweisungen zur Fehlerbehandlung, ist es besser, die Möglichkeiten zur Fehlerbehandlung zu nutzen.

Anwendungs-Datei:

1 Die On Error-Anweisung

Für die Möglichkeit, in VBA auftretentende Laufzeitfehler zu behandeln, gibt es die Fehleranweisung *On Error* und ein *Err*-Objekt. Die Anweisung *On Error* aktiviert eine Fehlerbehandlungsroutine und gibt die Position der Routine innerhalb einer Prozedur an. Die Routine wird auch als Errorhandler bezeichnet. Die Anweisung kann auch zum Deaktivieren eines Errorhandlers verwendet werden.

Die Syntax der Anweisung lautet

```
On Error GoTo Line
On Error Resume Next
On Error GoTo 0
```

On Error GoTo Line aktiviert den Errorhandler, der bei Line beginnt. Das Line-Argument ist eine beliebige Zeilenbezeichnung oder eine Zeilennummer. Tritt ein Laufzeitfehler auf, wird der Errorhandler aktiviert. Das Line-Argument muss in der gleichen Prozedur wie die *On Error*-Anweisung stehen, andernfalls tritt ein Compile-Time-Fehler auf.

On Error Resume Next bestimmt, dass beim Auftreten eines Laufzeitfehlers die auf die fehlerhafte Anweisung folgende Anweisung ausgeführt wird.

On Error GoTo 0 deaktiviert jeden aktivierten Errorhandler in der aktuellen Prozedur.

Um zu verhindern, dass Errorhandlercode ausgeführt wird, wenn kein Laufzeitfehler aufgetreten ist, sollte vor dem Line-Argument eine Exit Sub-, Exit Function- oder Exit Property-Anweisung stehen. Die Konstruktion eines solchen Fragments kann wie folgt aussehen.

```
Sub ProzedurName (Par1, Par2, …)
   On Error GoTo ErrorHandler
   Anweisungen
   Exit Sub
ErrorHandler:
   ErrorHandlerAnweisungen
   Resume Next
End Sub
```

Die Möglichkeiten der *On Error*-Anweisung zeigt das folgende Flussdiagramm (Bild 1).

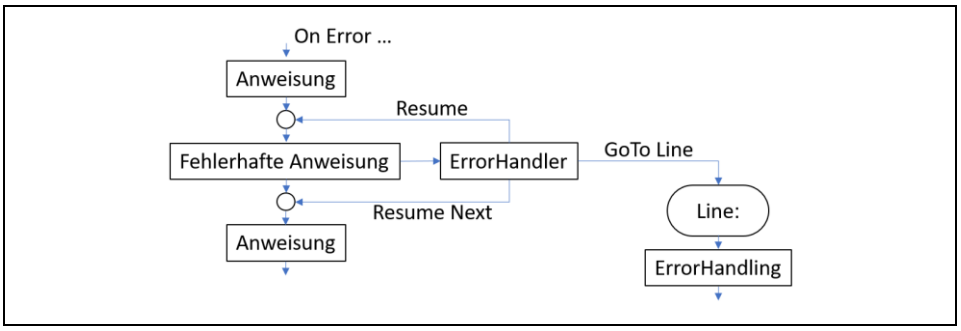

Bild 1. Fehlerhandling mit der On Error-Anweisung

2 Das Err-Objekt

Wenn ein Laufzeitfehler auftritt, dann werden die Eigenschaften des Err-Objekts mit Informationen gefüllt, die den Fehler eindeutig identifizieren. Das Err-Objekt ist ein intrinsisches Objekt mit globalem Bereich, sodass keine Instanz davon erstellt werden muss.

Die Syntax lautet

```
Err[.{Eigenschaft, Methode}]
```

Tabelle 1. Eigenschaften des Err-Objekts

Eigenschaft	Beschreibung
Description	Liefert einen beschreibenden Text, der einem Fehler zugeordnet ist
HelpContext	Liefert einen Zeichenfolgenausdruck, der die Kontext-ID für ein Thema in einer Hilfedatei enthält.
HelpFile	Liefert einen Zeichenfolgenausdruck mit dem vollqualifizierten Pfad zu einer Hilfedatei.
LastDllError	Liefert einen Systemfehlercode, der durch einen Aufruf einer Dynamic Link Library (DLL) erzeugt wird.
Number	Liefert die Fehlernummer und ist die Standardeigenschaft des Err-Objekts.
Source	Liefert einen Zeichenfolgenausdruck, der das Objekt darstellt, das den Fehler generiert hat.

Tabelle 2. Methoden des Err-Objekts

Methode	Beschreibung
Clear	Löscht alle Eigenschafts-Einstellungen des Err-Objekts.
Raise	Erzeugt einen Laufzeitfehler.

In der folgenden Prozedur wird ein Laufzeitfehler, Division durch Null, erzeugt und durch das Errorhandling werden Eigenschaften des Err-Objekts ausgegeben (Bild 2).

Codeliste 1. Anzeige von Err-Objekt-Eigenschaften

```
Sub ErrorInfo()
   Dim sInfo As String
   Dim dWert As Double
   On Error Resume Next

'Division durch Null
   dWert = 1 / 0

   If Err.Number <> 0 Then
      sInfo = "Err # " & Str(Err.Number) & _
         " ist aufgetreten in " & Err.Source & _
         vbCrLf & Err.Description
      MsgBox sInfo, vbInformation, "Error"
   End If
End Sub
```

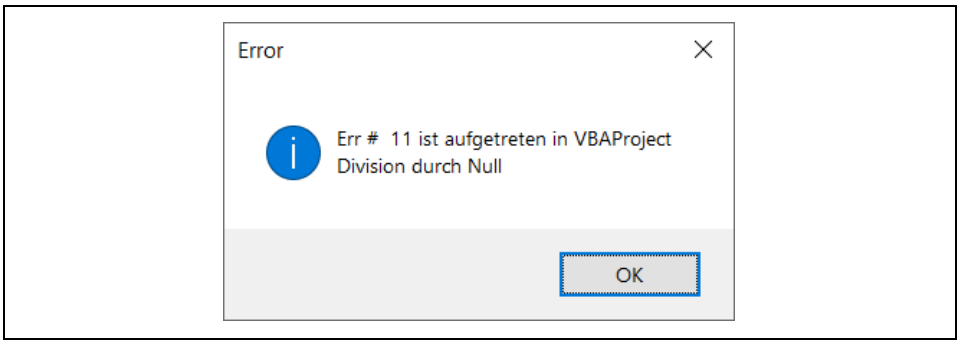

Bild 2. Infotext zum Laufzeitfehler

Die folgende Prozedur erzeugt mithilfe der *Raise*-Methode einen Laufzeitfehler und prüft im ErrorHandling, ob es dazu eine bezogene Beschreibung gibt, die dann im Direktfenster ausgegeben wird.

Codeliste 2. Integrierte Laufzeitfehler-Beschreibungen

```
Sub ErrorDescriptions()
   Dim iNum As Integer
   On Error GoTo ErrHandler

   For iNum = 0 To 750
      Err.Raise (iNum)
   Next
   Exit Sub

ErrHandler:
   If Err.Description <> _
      "Anwendungs- oder objektdefinierter Fehler" Then
      Debug.Print iNum, Err.Description
   End If
   Resume Next
End Sub
```

Am Ende eines ErrorHandlings sollte die Fehlernummer gelöscht werden, um zukünftige Probleme zu vermeiden. Die kann mit der *Clear*-Methode geschehen, aber auch mit der Anweisung *On Error GoTo -1*. Die Clear-Methode setzt den eigentlichen Fehler nicht zurück, sondern nur die Fehlernummer *Err.Number*.

In der folgenden Prozedur wird nur mit *On Error GoTo -1* der ErrHandler2 ausgeführt. Das lässt sich durch Auskommentierung leicht prüfen.

Codeliste 3. Laufzeitfehlerfolge

```
Sub ErrorSequence()
   On Error GoTo ErrHandler1:
   Error (7) 'nicht genügend Speicher
   Exit Sub

ErrHandler1:
   On Error GoTo -1 'Fehler löschen
'   Err.Clear
   On Error GoTo ErrHandler2:
   Error (16) 'Ausdruck zu komplex
```

```
   Exit Sub

ErrHandler2:
   Debug.Print Err.Description
End Sub
```

3 Fehlerfunktionen

Eine weitere Möglichkeit, mit Fehlern umzugehen, sind VBA-Funktionen die sich auf Fehler beziehen. Eine ist die boolesche VBA-Funktion *IsError*. Sie testet eine Formel, ob sie fehlerhaft (True) oder fehlerfrei (False) ist. Die folgende Prozedur testet einen Ausdruck in Zelle A1 einer Excel-Arbeitsmappe.

Codeliste 4. Anwendung von IsError

```
Sub IsErrorSample()
   MsgBox IsError(Range("A1").Value)
End Sub
```

Eine andere Möglichkeit ist die Verwendung der Worksheet-Funktion *IfError*. Sie liefert einen Wert, der angegeben wird, wenn eine Formel zu einem Fehler ausgewertet wird. Andernfalls ist es das Ergebnis der Formel. Im folgenden Beispiel wird die Formel in Zelle A1 getestet und liefert bei fehlerfreier Formel das Ergebnis der Formel, ansonsten wird der Text „Fehler!“ ausgegeben.

Codeliste 5. Anwendung der WorksheetFunction IfError

```
Sub IfErrorSample()
   Dim vReturn As Variant
   vReturn = WorksheetFunction. _
      IfError(Range("A1").Value, "Fehler!")
   MsgBox vReturn
End Sub
```