



## Mit Fehlern in Access umgehen

PAUL ROHORZKA  
paul.rohorzka@techtalk.at  
AEK14, Oktober 2011

COPYRIGHT 2011, TECHTALK - WWW.TECHTALK.AT

---

---

---

---

---

---

---

---



## Fehler in Access – eine Einführung

Wo Fehler herkommen  
Wie Fehler entstehen

COPYRIGHT 2011, TECHTALK - WWW.TECHTALK.AT

---

---

---

---

---

---

---

---

### Programmtechnische Fehlerquellen (1)

- Daten-**Validierung**
  - Text zu lang
  - Erforderliches Feld leer gelassen
  - Zahl außerhalb des Wertebereichs
  - Gültigkeitsregel verletzt
- Access: **Eingabeformat** passt nicht
  - Datum/Uhrzeit
  - Währung
  - Benutzerdefiniertes Eingabeformat




---

---

---

---

---

---

---

---

### Programmtechnische Fehlerquellen (2)

- Laufzeitfehler im Code
  - Bibliotheken (DAO, ADODB, Office, ...)
  - DB weg, Dokument kann nicht gespeichert werden
- Eigener Code
  - Programmfehler
  - Division durch 0, nicht festgelegte Objektreferenz, ...
  - Bewusst ausgelöste Fehler
- Falsches Backend
  - z.B. alte Datenstruktur




---

---

---

---

---

---

---

---

### Andere Fehler

- Probleme mit **Infrastruktur**
  - Datei/Backend nicht verfügbar
  - Festplatte voll
  - Timeout (System ausgelastet)
  - ...
- **Bedienfehler**
  - Der Benutzer benützt die Anwendung nicht wie erwartet
- Probleme mit der **Funktionalität**
  - Die Anwendung hilft dem Benutzer nicht
  - Vortrag Geschäftsregeln




---

---

---

---

---

---

---

---

### Das Ereignis Form\_Error

- Tritt beispielsweise ein
  - wenn Access eine Eingabe nicht akzeptiert (falsches Eingabeformat)
  - wenn vom Backend Fehler kommen (nur DAO)
- Nicht bei VBA-Fehlern!




---

---

---

---

---

---

---

---

### Parameter von Form\_Error

```
Form_Error(DataErr As Integer, _
           Response As Integer)
```

- **DataErr:** Fehlernummer
  - Meldung mit Platzhaltern und Formatierung mittels AccessError(DataErr)
- **Response:** Wie geht's weiter?
  - acDataErrDisplay: → Standardmeldung
  - acDataErrContinue: → Keine Standardmeldung




---

---

---

---

---

---

---

---

### Wofür Form\_Error einsetzen?

- Klarere Fehlermeldungen
- Fehler protokollieren
- Aber auch: Sondereingaben zulassen
  - Idee: © Henry Habermacher  
<http://www.dbdev.org/down31.htm>




---

---

---

---

---

---

---

---




---

---

---

---

---

---

---

---

### Genese eines Laufzeitfehlers

WTF?

Access  
Anwendung  
Excel  
XLS-Datei

Umsätze neu berechnen

```

Private Sub btnNeuberechnen_Click()
    UmsatzlisteAktualisieren 2011
End Sub

Private Sub UmsatzlisteAktualisieren(Jahr As Integer)
    ...
    ... = MyWorkbook.Sheets("Umsätze " & Jahr)
End Sub
    
```

Microsoft Visual Basic for Applications - VBE

Run-time error '9':  
Subscript out of range

OK Help

Es gibt kein Tabellenblatt [Umsätze 2011]!

TECHTALK

---

---

---

---

---

---

---

---

### Was wir lernen können

- Fehler möglichst früh erkennen!
  - Mehr und konkreter Kontext verfügbar
  - → Nicht auf Folgefehler verlassen
- Fehlermeldungen aus Komponenten
  - sind meistens zu technisch
  - fehlen oft wichtige Informationen
  - → Fehlermeldungen selbst in die Hand nehmen

TECHTALK

---

---

---

---

---

---

---

---

### Debuggen mit Fehlern

- Call Stack anzeigen in Breakpoint (Ctrl+L)

- Verhalten bei Fehlern steuern (Option, nur MDB)

Error Trapping

- Break on all errors
- Break in Class Module
- Break on Unhandled Errors

- Per Code:  
Access.SetOption(„Error Trapping“, 0|1|2)

TECHTALK

---

---

---

---

---

---

---

---




---

---

---

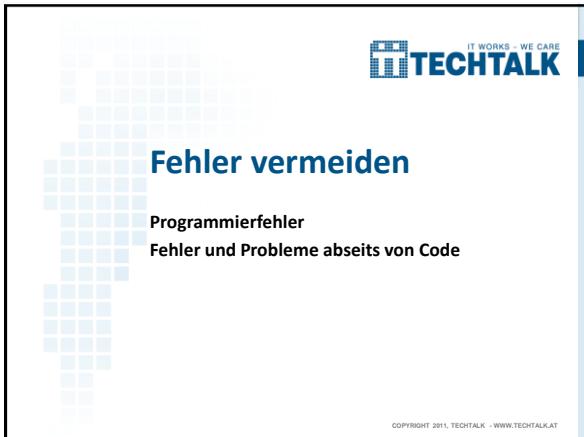
---

---

---

---

---




---

---

---

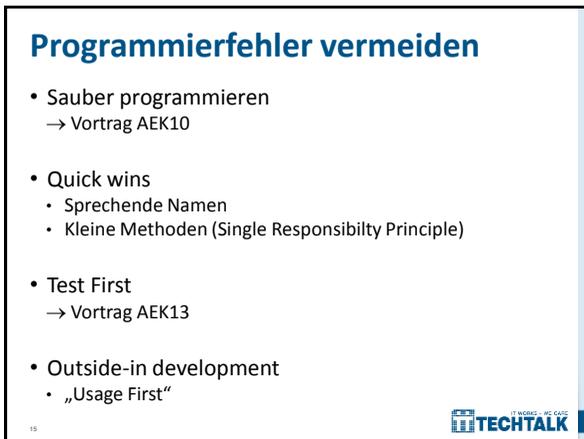
---

---

---

---

---




---

---

---

---

---

---

---

---




---

---

---

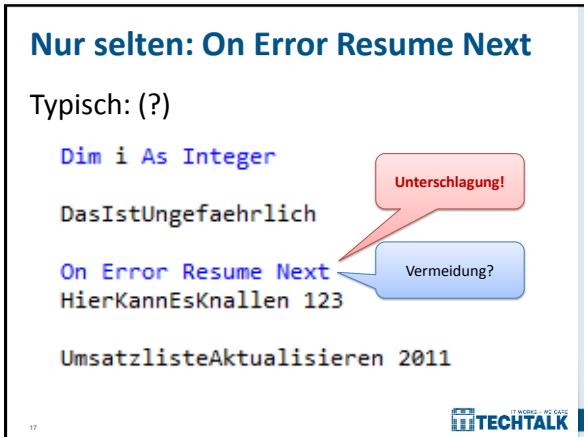
---

---

---

---

---




---

---

---

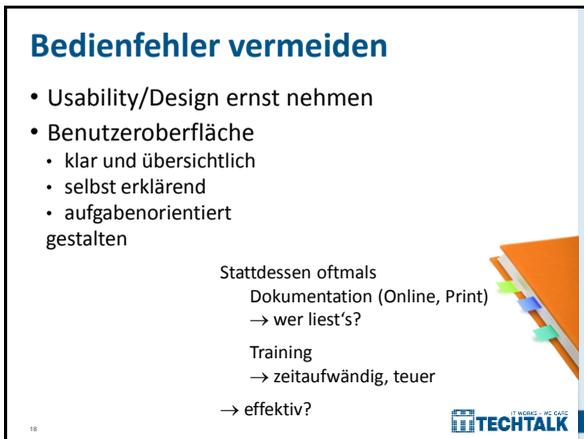
---

---

---

---

---




---

---

---

---

---

---

---

---

### Funktionale Fehler vermeiden

- Anforderungen
  - mit Endbenutzer erarbeiten
  - gemeinsam Kriterien festlegen
  - rasch umsetzen und ausliefern

→ tlw. Vortrag Geschäftsregeln




---

---

---

---

---

---

---

---

### Mit Infrastrukturproblemen umgehen

#### Vermeiden?

- Durch die Anwendung nicht möglich



#### Wie dann?

- Technisch durch Administration
- Organisatorisch

Möglich ist aber:  
**Testen** wie sich das Programm im Fehlerfall verhält!




---

---

---

---

---

---

---

---

### Alles perfekt?

- Wir können nicht alle Fehler vermeiden
- Versteckte Fehler herausfordern!
  - Testen: manuell, automatisiert
  - → Vortrag AEK13
- Mit verbliebenen Fehlern rechnen
  - → Fehlerbehandlung
- Fehler außerhalb der Anwendung nicht ignorieren




---

---

---

---

---

---

---

---

**Zum Mitnehmen für Dich!**

- ♥ Viele Fehler sind vermeidbar
- ♥ Versteckte Fehler herausfordern
- ♥ Mit verbliebenen Fehlern rechnen



**TECHTALK** IT WORKS - WE CARE

---

---

---

---

---

---

---

---

**TECHTALK** IT WORKS - WE CARE

## Laufzeitfehler behandeln

Anatomie eines Laufzeitfehlers  
 Programmfluss steuern  
 Benutzerinteraktion

COPYRIGHT 2011, TECHTALK - WWW.TECHTALK.AT

---

---

---

---

---

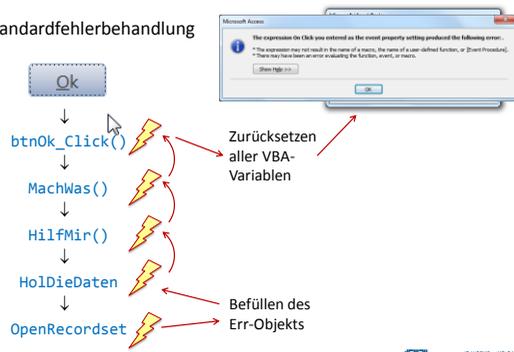
---

---

---

## Was passiert bei einem Laufzeitfehler?

Standardfehlerbehandlung



**TECHTALK** IT WORKS - WE CARE

---

---

---

---

---

---

---

---

### Das Err-Objekt

- wird bei Laufzeitfehlern befüllt
- Members
  - Description
  - Number
  - Source
  - HelpSource/HelpContext
- blubbert den CallStack nach oben bis
  - Access Standard-ErrorHandler
  - Manueller ErrorHandler (On Error ...)
- sein Zustand geht verloren
  - bei On Error ...
  - nach ErrorHandler
  - wenn ein weiterer Fehler geworfen wird
  - wenn das Programm zurückgesetzt wird




---

---

---

---

---

---

---

---

### Programmfluss im Fehlerfall steuern (1)

#### 1. Frage: Was tun im Fehlerfall?

`On Error Goto <Label>`

→ Im Fehlerfall weiter bei <Label>

`On Error Resume Next`

→ Ab jetzt keine Fehler auslösen

SOP:  
Immer paarweise mit  
`On Error Goto 0!`

`On Error Goto 0`

→ Standardfehlerbehandlung wiederherstellen




---

---

---

---

---

---

---

---

### Vorsicht, Mumpitz!



*Rein da!*

„Zur Sicherheit  
`On Error Resume Next`  
davor schreiben.“

**Nein!**

Ich will wissen wenn es  
kracht!




---

---

---

---

---

---

---

---

## Programmfluss im Fehlerfall steuern (2)

2. Frage: Was tun nach der Fehlerbehandlung?

**Resume**

→ Vorsicht Endlosschleife! (Retry-Counter)

**Resume Next**

→ Egal, machen wir trotzdem weiter

**Resume <Label>**

→ Weiter geht's bei <Label> (Aufräumen)




---

---

---

---

---

---

---

---

## Klassischer Errorhandler

```

Sub MySub()
On Error Goto ErrLabel
' WorkCode - ev. tritt Fehler auf
ExitLabel:
' CleanupCode
Exit Sub
ErrLabel:
MsgBox Err.Description
Resume ExitLabel
End Sub
    
```




---

---

---

---

---

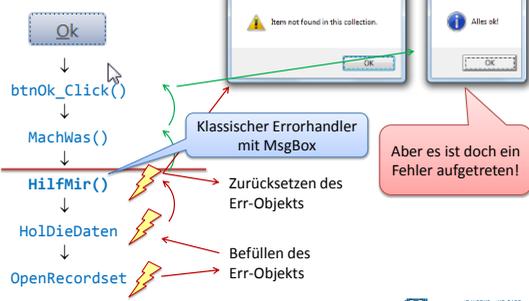
---

---

---

## Was passiert bei einem Laufzeitfehler?

Mit klassischem Errorhandler




---

---

---

---

---

---

---

---

### Nachteile eines Errorhandlers

- Frist bestehende Fehler
  - Vorsicht beim Aufräumen/Loggen
- Verschleiert ev. Probleme
- Macht die Methode unübersichtlich

→ **Konsequenz:**  
 Fehlerbehandlung  
 a) **vermeiden** (sic!)  
 b) nur **begründet** einsetzen




---

---

---

---

---

---

---

---

### Vorsicht, Mumpitz!




---

---

---

---

---

---

---

---

### Fehlerbehandlung vermeiden: Beispiel

Element aus Collection liefern oder Nothing

```
• Typisch:
  On Error Resume Next
  Set ReturnValue = m_Coll(Key)
  On Error Goto 0
```

```
• Variante ohne Fehlerbehandlung:
  For Each Item In m_Coll
  If Item.Key = Key Then
  Set ReturnValue = Item
  Exit Function
  End If
  Next
```

Sorge um Performance?  
 → Zeig mir eine Messung!




---

---

---

---

---

---

---

---

### Wann eine Fehlerbehandlung? (1)

#### 1. Ressourcen wieder freigegeben

- Datenbank, Recordset, Anwendung, Datei, ...

Beispiel:

```
Dim Rst As DAO.Recordset
Set Rst = CurrentDb.OpenRecordset(...)
...
Cleanup:
If Not Rst Is Nothing Then Rst.Close
Set Rst = Nothing
```




---

---

---

---

---

---

---

---

### Wann eine Fehlerbehandlung? (2)

#### 2. Sonderbehandlung bestimmter Fehler

- Eine Alternative versuchen
- Fehlermeldung konkretisieren

Beispiel:

```
...
Err_:
If Err.Number = 1234 Then ' File not found
... "Das Backend in "" & BackendPath & _
... "" existiert nicht."
End If
...

```




---

---

---

---

---

---

---

---

### Wann eine Fehlerbehandlung? (3)

#### 3. Informationen über die Entstehung des Fehlers sammeln

- Zustandsinformationen (Parameter, Variablen)
- ~CallStack aufbauen ([Err.Source?](#))

Beispiel:

```
...
Err_:
... Err.Source & ";" & _
... TypeName(Me) & ".MyMethod" ...
```




---

---

---

---

---

---

---

---

### Unser Dilemma

- Angenommen:  
Wir brauchen eine Fehlerbehandlung
- Die aufrufende Prozedur muss den Fehler trotzdem mitbekommen

→ Nach der Behandlung den Fehler (erneut) werfen  
**Err.Raise**




---

---

---

---

---

---

---

---

### Klassischer ErrorHandler reloaded

Klassisch

```

Sub MySub()
On Error Goto ErrLabel
' WorkCode - ev. tritt Fehler auf
ExitLabel:
' CleanupCode
Exit Sub
ErrLabel:
' Fehler anzeigen/protokollieren
Err.Raise Err.Number
Resume ExitLabel
End Sub

```

Aber wer räumt jetzt auf?




---

---

---

---

---

---

---

---

### Vorschlag für ErrorHandler

```

Sub MySub()
On Error Goto Exit_
' WorkCode - ev. tritt Fehler auf
Exit_:
Select Case Err.Number
Case 0 ' Nichts passiert
Case SomeSpecialErrorNumber
' Mach was Spezielles
Case Else ' Unerwarteter Fehler
' Logging
Gosub Cleanup
Err.Raise ...
End Select
Gosub Cleanup
Exit Sub
Cleanup:
' CleanupCode
Return
End Sub

```




---

---

---

---

---

---

---

---




---

---

---

---

---

---

---

---




---

---

---

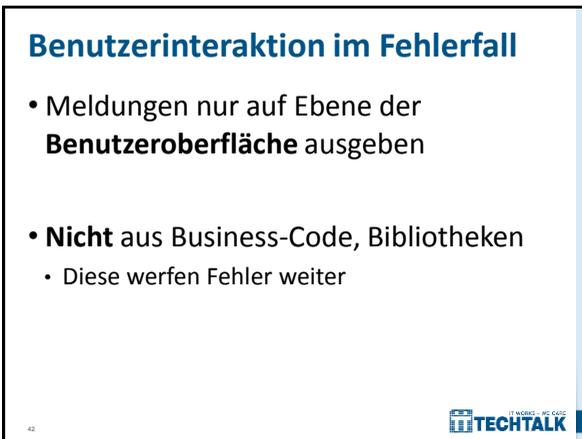
---

---

---

---

---




---

---

---

---

---

---

---

---

## Gestalten von Fehlertexten

- **Konsistent** formulieren
  - Aktiv vs. passiv
- **Konkret** formulieren
  - Ev. am Weg nach oben umformulieren
- Hinweise zur **Behebung/Vermeidung** geben
- Trotzdem möglichst **kurz** halten
  - Ev. Details optional anzeigen

43




---

---

---

---

---

---

---

---

## Fehlermeldungen anzeigen

- Buttonbeschriftungen zum Text passend wählen
- Default Button sinnvoll wählen
- Eventuell MsgBox ersetzen?
  - Konkrete Texte statt Ok/Abbrechen oder Ja/Nein
  - Details ein-/ausblenden
  - Protokollieren

44




---

---

---

---

---

---

---

---

## vbWatchdog (kommerziell)

- Aufruf eines globalen Errorhandlers
- Zugriff auf
  - Call Stack
  - Zeilennummern (nicht im Code!)
  - Inhalt von Variablen, Parametern, ...
- Frei gestaltbarer Fehlerdialog (HTML)
- Bestimmung wie's danach weitergeht
- Kompakte Fehlerbehandlung mit Try/Catch

45




---

---

---

---

---

---

---

---

**Zum Mitnehmen für Dich!**

- ♥ Fehlerbehandlung bewusst einsetzen
- ♥ Kommunikation mit Benutzer im Fehlerfall ernst nehmen

**TECHTALK** IT WORKS - WE CARE

---

---

---

---

---

---

---

---

**TECHTALK** IT WORKS - WE CARE

*und andere interessante Dinge*

**Fehlerprotokollieren**

- Warum?
- Was?
- Wo?
- Wohin?
- Wie?
- Was dann?

COPYRIGHT 2011, TECHTALK - WWW.TECHTALK.AT

---

---

---

---

---

---

---

---

**Warum protokollieren? (1)**

- **Hintergründe** zu Fehlerbeschreibungen bekommen
  - Vorgeschichte 
  - Interna
- **Fehler/Probleme** auch ohne Berichten von Benutzern **erkennen**
  - Z.B. häufig nicht beendete Vorgänge

48

**TECHTALK** IT WORKS - WE CARE

---

---

---

---

---

---

---

---

### Warum protokollieren? (2)

- Analyse des **Benutzerverhaltens**
  - Häufig/selten genutzte Funktionen
  - Was dauert wir lange?
  - Performance
- **Dokumentation**
  - Wer hat wann was gemacht?

Gesetzliche und betriebliche **Rahmenbedingungen** beachten!




---

---

---

---

---

---

---

---

### Was protokollieren? (1)

- **Umgebung** der Applikation
  - Versionen (genau – SPs!)
    - Windows, Office, DAO, ADO, ...
  - Performance Counter
    - Speicher, CPU-Auslastung, ...




---

---

---

---

---

---

---

---

### Was protokollieren? (2)

- **Lebenszyklus** der Applikation
  - Session
    - User (Windows/Access/Applikation)
    - PC (Name, IP, ...)
    - Zeit Start/Ende
  - Bibliotheksverweise
    - IsBroken?
  - Objektinstanzen
    - insgesamt/gleichzeitig/noch offen




---

---

---

---

---

---

---

---

### Was protokollieren? (3)

- Programmfluss
  - Funktionsaufrufe Beginn/Ende/Fehler
- Details
  - Was auch immer gerade von Interesse ist

52




---

---

---

---

---

---

---

---

### Wo protokollieren?

- Neuralgische Punkte (vorher/nachher)
- Methodenbeginn/Ende
- Allgemeine Blöcke (Beginn/Ende)
- Potenzielle Fehlerstellen

53




---

---

---

---

---

---

---

---

### Wohin protokollieren?

- Datenbank
  - Nicht für grundlegende technische Probleme
- Datei
  - Eine Zeile pro Eintrag
- EventLog
  - Läuft leicht über
  - Admins gut bekannt
- Mail
  - Für spezielle Ereignisse
  - An PowerUser, Administrator, Entwickler

54




---

---

---

---

---

---

---

---

### Wie protokollieren?

- Zentraler Logger
- Ziel des Protokolls (Datei/Datenbank/...) **zentral** konfigurieren
- Keine Fehlerbehandlung beim Loggen eines Fehlers!
- Automatische Logging des Endes:
 

```
With Logger.LogBlock("Excel Export")
  ...
End With
```




---

---

---

---

---

---

---

---




---

---

---

---

---

---

---

---

### Protokolle auswerten

- Datenbank
  - SQL
  - Access
- Datei
  - Texteditor (Tipp: Notepad++)
    - Suchen, Spaltenoperationen, reguläre Ausdrücke, ...
  - Excel
  - Access
- Allgemein
  - Log-Analyse Tools




---

---

---

---

---

---

---

---

**Zum Mitnehmen für Dich!**

- ♥ Kenne das Verhalten deiner Anwendung in Produktion
- ♥ Ziehe aus den gesammelten Daten Schlüsse & Konsequenzen



IT WORKS - WE CARE  
**TECHTALK**

---

---

---

---

---

---

---

---

**Für den Weg: Kenne deine Fehler!**

-  Die schönsten Fehler sind die **vermiedenen**.
-  Die besten Fehler sind die **aufschlussreichen**.
-  Die schlimmsten Fehler sind die **unbemerkten**.

IT WORKS - WE CARE  
**TECHTALK**

---

---

---

---

---

---

---

---

IT WORKS - WE CARE  
**TECHTALK**

**Und jetzt – ran an den Fehler!**

PAUL ROHORZKA  
paul.rohorzka@techtalk.at  
AEK14, Oktober 2011

COPYRIGHT 2011, TECHTALK - WWW.TECHTALK.AT

---

---

---

---

---

---

---

---