Excel + VBA

Sequentielle Textdateien

Autor & Copyright: Dipl.-Ing. Harald Nahrstedt

Version: 2016 / 2019 / 2021 / 365

Erstellungsdatum: 12.02.2012

Überarbeitung: 01.12.2023

Beschreibung:

In Textdateien lassen sich schnell Informationen speichern und auch wieder schnell abrufen. Dabei sind diese Dateien für ein Schreiben und Lesen des gesamten Textes gedacht. Eben ein sequentielles Schreiben und Lesen von Textzeilen. Andere Methoden lassen sich mit einem kleinen Programmieraufwand auch bewerkstelligen. Die Handhabung der Daten geht bei großen Datenmengen aber deutlich nur mit großem Zeitaufwand.

Anwendungs-Datei: AE-018_SequentielleTextdateien.xlsm

1 Struktur einer sequentiellen Datei

In Textdateien lassen sich schnell Informationen speichern und wieder abrufen. Dies trifft nur für den gesamten Inhalt einer Textdatei zu. Das liegt daran, dass die Zeilen einer Textdatei nur sequentiell hintereinander in eine Textdatei geschrieben, bzw. gelesen werden können. Daher auch die Bezeichnung *sequentielle Datei*. Eine sequentielle Datei hat folgende Struktur

1. Zeile: Eine beliebige Folge von Textzeichen CR LF
2. Zeile: Eine beliebige Folge von Textzeichen CR LF
3. …

In einer Zeile steht eine beliebige Anzahl Buchstaben, Ziffern und Sonderzeichen. Die Zeile endet mit zwei Steuerzeichen. Nämlich CR für Carriage Return (VBA-Konstante vbCrb, ASCII-Wert = 13) und LF für Line Feed (VBA-Konstante vbLF, ASCII-Wert = 10). Beide Steuerzeichen sind in der VBA-Konstanten vbCRLF vereint. Die Bezeichnungen stammen noch aus einer Zeit, als jedermann eine mechanische Schreibmaschine benutzte. Hatte man eine Zeile geschrieben und wollte man eine neue Zeile beginnen, dann musste man einen Hebel nach rechts schieben, dadurch wurde der Wagen (am Hebel eine Anschlagwalze mit eingespannten Blatt) nach rechts geschoben (Carriage Return) , also an den Zeilenanfang, und gleichzeitig wurde die Walze mit dem eingespannten Blatt um eine Zeile nach oben gedreht, also eine neue Zeile begonnen (Line Feed). In Word lässt sich ein Modus einstellen, der diese Steuerzeichen anzeigt.

2 In Textdateien schreiben und lesen

Die übliche Handhabung einer Textdatei sieht folgende Vorgehensweise vor. Den gesamten Text einer sequentiellen Datei in eine Variable lesen, Änderungen durchführen und dann den gesamten Text wieder speichern.

Die nachfolgende Prozedur schreibt einen aus mehreren Zeilen bestehenden Text (in einer Variablen) in eine sequentielle Datei. Damit der Inhalt einer bestehenden Datei gleichen Namens verloren geht, wird die Datei im *Output-Modus* geöffnet. Eine weitere Prozedur liest den Inhalt der so erstellten Datei. Wir erstellen ein Modul, das die Prozeduren und Daten einer Sequentiellen Datei enthält (Bild 1).

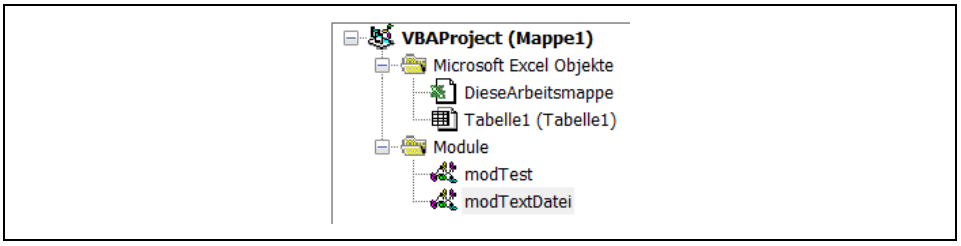

Bild 1. Verwendete Module

Codeliste 1. Prozeduren zur Handhabung einer sequentiellen Datei im Modul modTextDatei

```
Option Explicit

'Text in Datei speichern, alter inhalt geht verloren
```

```
Public Sub WriteAllText(ByVal sFilename As String, _
   ByVal sLines As String)
   Dim iFile As Integer

   iFile = FreeFile
   Open sFilename For Output As #iFile
   Print #iFile, sLines
   Close #iFile
End Sub

'Lesen des gesamten Inhaltes einer Textdatei
Public Function sReadAllText(ByVal sFilename As String) _
   As String
   Dim iFile   As Integer

   'Existiert die Datei ?
   If Dir(sFilename, vbNormal) <> "" Then
      'Textdatei im Binärmodus öffnen und gesamten
      'Inhalt in einem Rutsch auslesen
      iFile = FreeFile
      Open sFilename For Binary As #iFile
      sReadAllText = Space$(LOF(iFile))
      Get #iFile, , sReadAllText
      Close #iFile
   End If
End Function
```

Die Prozeduren zum Testen der Datei-Prozeduren schreiben wir in ein separates Code-Modul. Zuerst wird mit der Prozedur *TextDateiSchreiben* eine Testdatei erzeugt.

Codeliste 2. Testprozeduren zum Schreiben und Lesen einer sequentiellen Textdatei im Modul modTest

```
Option Explicit

Public Const sFilename = "C:\Temp\Test.txt"

Sub TextDateiSchreiben()
   Dim sLines     As String

   sLines = "Dies ist ein Testtext!" & vbCrLf
   sLines = sLines & "Dies ist die zweite Zeile." & vbCrLf
   sLines = sLines & "Dies ist die dritte Zeile." & vbCrLf
   sLines = sLines & "Dies ist die vierte Zeile." & vbCrLf
   sLines = sLines & "Dies ist die fünfte Zeile." & vbCrLf
   sLines = sLines & "Dies ist die letzte Zeile."
   Call WriteAllText(sFilename, sLines)
End Sub

Sub TextDateiLesen()
   MsgBox sReadAllText(sFilename)
End Sub
```

Nach Ausführung der Prozedur *TextDateiLesen* zeigt diese in einer MsgBox den Inhalt der Datei (Bild 2).

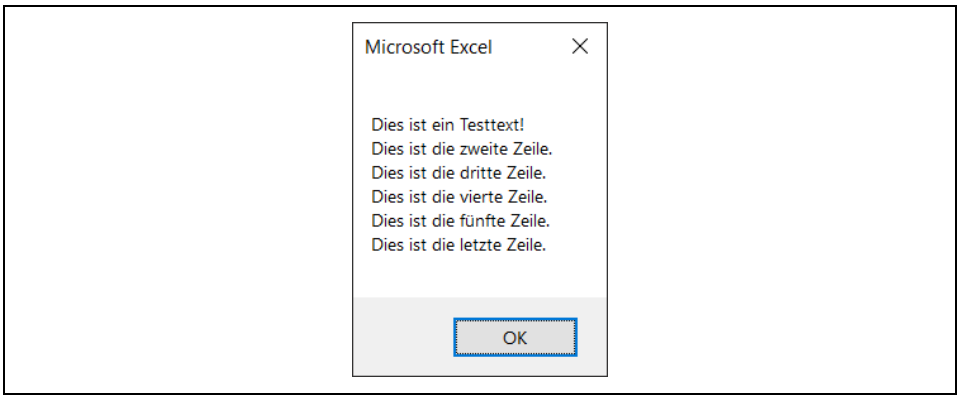

Bild 2. Inhalt der Textdatei in einer MsgBox

3 Textdateien Text anfügen

Einer bestehenden Textdatei weitere Zeilen anzuhängen, ist im Dateimodus *Append* sehr einfach. Die nachfolgende Datei-Prozedur macht es möglich.

Codeliste 3. Die Prozedur im Modul modTextDatei fügt Text zu einer bestehenden Textdatei hinzu

```
'Zeilen an Textdatei anhängen
Public Sub AppendLineText(ByVal sFilename As String, _
   ByVal sLines As String)
   Dim iFile As Integer

   iFile = FreeFile
   Open sFilename For Append As #iFile
   Print #iFile, sLines
   Close #iFile
End Sub
```

Die entsprechende Testprozedur fügt zwei weitere Zeilen an. Zu beachten ist, dass vbCrLf nur zwischen die Zeilen gesetzt werden muss.

Codeliste 4. Die Prozedur in Modul modTest fügt zwei weitere Zeilen hinzu

```
Sub TextDateiErweitern()
   Dim sLines As String

   sLines = "Dies ist die erste neue Zeile." & vbCrLf
   sLines = sLines & "Dies ist die zweite neue Zeile."
   Call AppendLineText(sFilename, sLines)
End Sub
```

Der Inhalt wird durch erneutes Lesen überprüft (Bild 3).

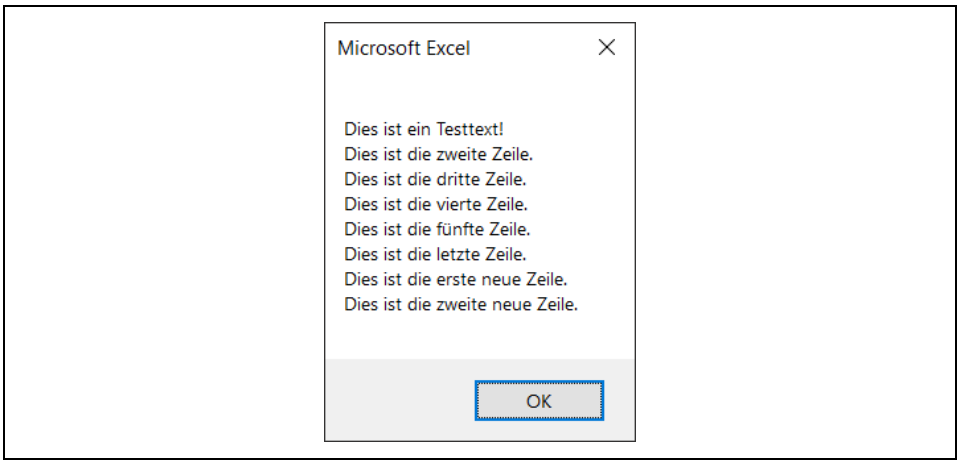

Bild 3. Neuer Inhalt der Textdatei

4 Bestimmte Zeilen aus Textdateien lesen

Möchte man eine bestimmte Zeile aus der Textdatei verwenden, so muss die nachfolgende Datei-Prozedur *sReadLineText* verwendet werden. Sie liest alle Zeilen der Textdatei und zählt sie, bis die entsprechende Zeile gelesen wird.

Codeliste 5. Die Prozedur in Modul modTextDatei liest eine bestimmte Zeile aus der Textdatei

```
' Lesen einer bestimmten Zeile einer Textdatei
Public Function sReadLineText(ByVal sFilename As String, _
   ByVal LineToRead As Long) As String
   Dim iFile   As Integer
   Dim sLine   As String
   Dim lRow    As Long

   lRow = 0
   'Existiert die Datei ?
   If Dir(sFilename) <> "" Then
      iFile = FreeFile
      Open sFilename For Input As #iFile
      'Solange einlesen, bis entweder Dateiende
      'oder gewünschte Zeilennummer erreicht
      While Not EOF(iFile) And lRow < LineToRead
         lRow = lRow + 1
         Line Input #iFile, sReadLineText
      Wend
      If EOF(iFile) Then sReadLineText = ""
      Close #iFile
   End If
End Function
```

Die nachfolgende Prozedur testet diese Funktion.

Codeliste 6. Die Prozedur in Modul modTest liest die fünfte Zeile aus der Textdatei

```
Sub TextZeile5Lesen()
   MsgBox sReadLineText(sFilename, 5)
End Sub
```

Die Ausführung liefert den fünften Satz in der Textdate (Bild 4).

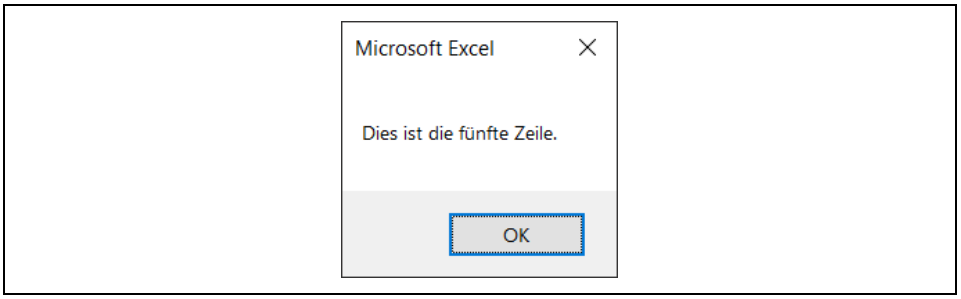

Bild 4. Die fünfte Textzeile

5 Textdateien ändern

Etwas komplizierter wird es, will man eine Zeile ersetzen oder einfügen. In diesem Fall gibt es mehrere Möglichkeiten.

- Wenn die Datei noch nicht existiert, muss diese und die Zeilen vor der Position als Leerzeilen angelegt werden.
- Existiert die Datei, aber die Position liegt hinter dem Dateiende, dann müssen ebenfalls Leerzeilen bis zur Einfügeposition angelegt werden.
- Ist die Zeile in der Datei vorhanden, kann sie durch die neue Zeile ersetzt oder hinter der neuen Zeile eingefügt werden.

Dabei wird eine temporäre Datei angelegt, in welcher der neue Inhalt der Quelldatei erstellt wird. Erst danach wird die Quelldatei gelöscht und anschließend durch eine Kopie der temporären Datei ersetzt. Im Modul *modTest* wird die temporäre Datei im Kopfbereich definiert.

```
Public Const sFileTemp = "C:\Temp\Temp.txt
```

Mithilfe der nun vorhandenen Datei-Prozeduren kann die gewünschte Datei-Prozedur zum Einfügen oder Ersetzen eine Textzeile in einer sequentiellen Datei erstellt werden.

Codeliste 7. Die Prozedur in Modul modTextDatei schreibt die fünfte Zeile in die Datei

```
'Einzelne Zeile in eine Textdatei speichern
Public Sub WriteLineText(ByVal sFilename As String, _
   ByVal LinePos As Long, ByVal sNewLine As String)
   Dim iFile      As Integer
   Dim iTemp      As Integer
   Dim lCount     As Long
   Dim lRow       As Long
   Dim sLine      As String

   If Dir(sFilename, vbNormal) = "" Then
      'Datei existiert nicht
```

```
      iFile = FreeFile
      Open sFilename For Output As #iFile
      'vorherige Leerzeilen erzeugen
      For lCount = 1 To LinePos - 1
         Print #iFile, ""
      Next lCount
      'Zeile speichern
      Print #iFile, sLine
      Close #iFile
   Else
      'Temporäre Datei erstellen
      'Quelldatei zum Lesen und
      'temporäre Datei zum Schreiben öffnen
      iFile = FreeFile: Open sFilename For Input As #iFile
      iTemp = FreeFile: Open sFileTemp For Output As #iTemp
      'Original-Datei einlesen und x. Zeile durch
      'neuen Inhalt ersetzen
      lRow = 0
      Do While Not EOF(iFile)
         lRow = lRow + 1
         Line Input #iFile, sLine
         If lRow = LinePos Then
            ' x. Zeile durch neuen Inhalt ersetzen
            sLine = sNewLine
         End If
         Print #iTemp, sLine
      Loop
      'notfalls müssen Leerzeilen ergänzt werden
      If lRow < LinePos Then
         For lCount = lRow + 1 To LinePos - 1
            Print #iTemp, ""
         Next lCount
         Print #iTemp, sNewLine
      End If
      'Dateien schliessen
      Close #iFile
      Close #iTemp
      'Alte Datei löschen
      Kill sFilename
      'temporäre Datei kopieren
      FileCopy sFileTemp, sFilename
      'temporäre Datei löschen
      Kill sFileTemp
   End If
End Sub
```

Und auch dafür wird eine Testprozedur benötigt.

Codeliste 8. Die Prozedur in Modul modTest überschreibt die fünfte Zeile in der vorhandenen Datei

```
Sub TextZeileEinfügen()
   Dim sLine      As String

   sLine = "Dies ist eine eingefügte Zeile."
   Call WriteLineText(sFilename, 5, sLine)
End Sub
```

Nach der Ausführung zeigt die Prozedur TextDateiLesen das Ergebnis (Bild 5).

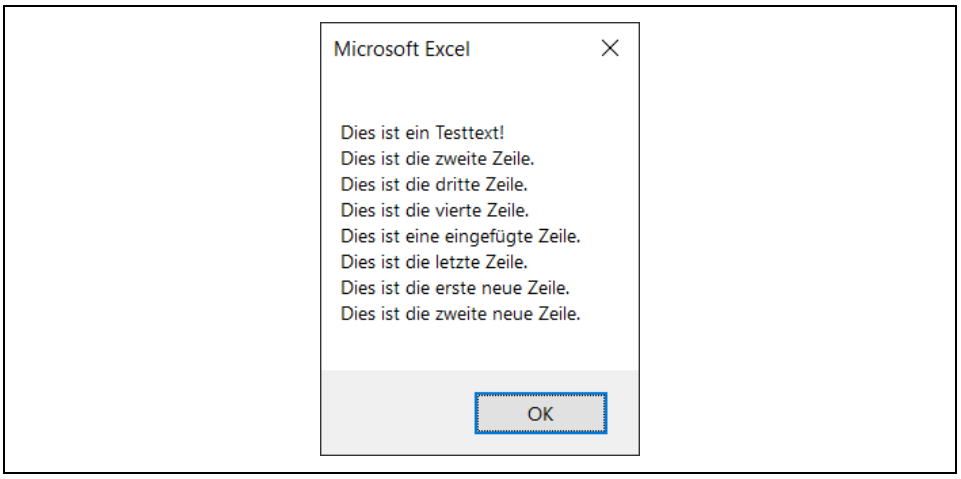

Bild 5. Neuer Inhalt der Textdatei mit eingefügter Zeile