BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Wirtschaftsinformatik 1
LE 09 – Zugriff auf das Dateisystem

Prof. Dr. Thomas Off

http://www.ThomasOff.de/lehre/beuth/wi1

Einordnung

06 – Zusammenfassung

05 – Debugger und Testen

04 – Fortgeschrittene Konzepte

04.A	04.B	04.C
Oberflächen (Teil 1) Elemente und Eigenschaften	Oberflächen (Teil 2) Ereignisverarbeitung	Zugriff auf Dateisystem und Anwendungen

03 – Grundkonzepte

03.A	03.B	03.C	03.D	03.E
Wert Ausdruck Variable Konstante Datentyp	Bedingte Ausführung/ Verzwei-gungen	Schleifen	Felder Mengen	Prozedur Funktion Modul

02 – Grundlagen der Programmierung

01 – Grundbegriffe der Wirtschaftsinformatik

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

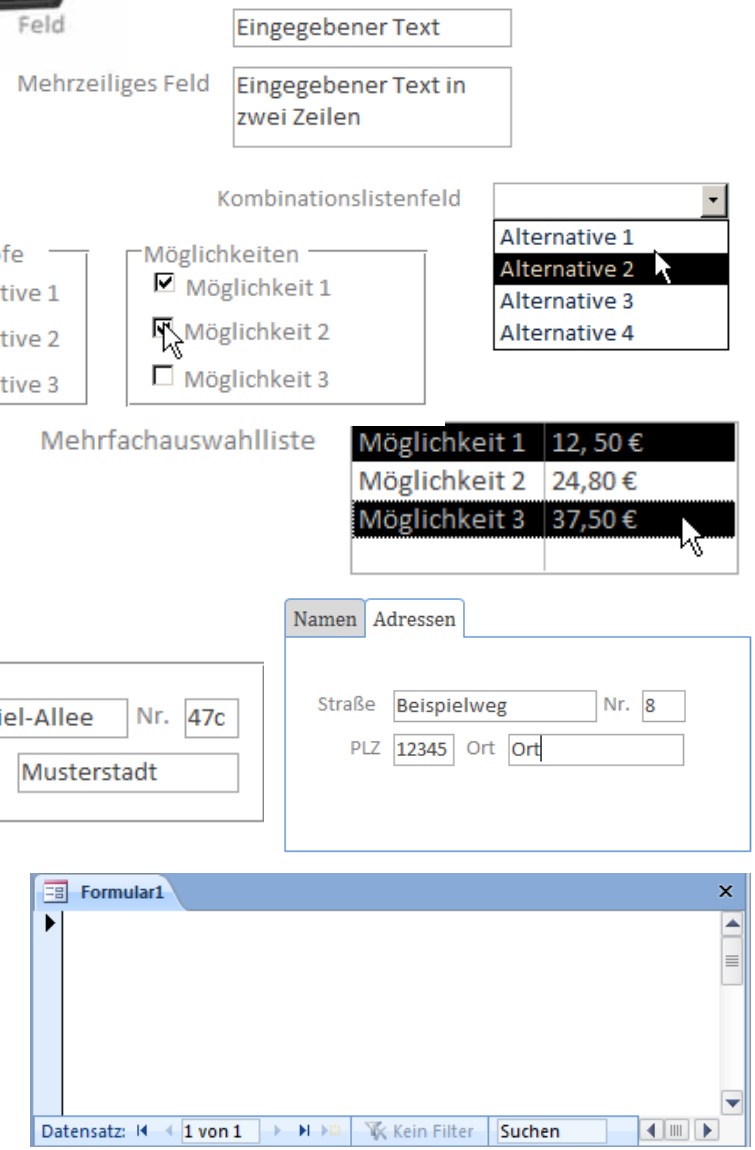
Inhalt

Rückblick

Rückblick

Wichtige Oberflächenelemente

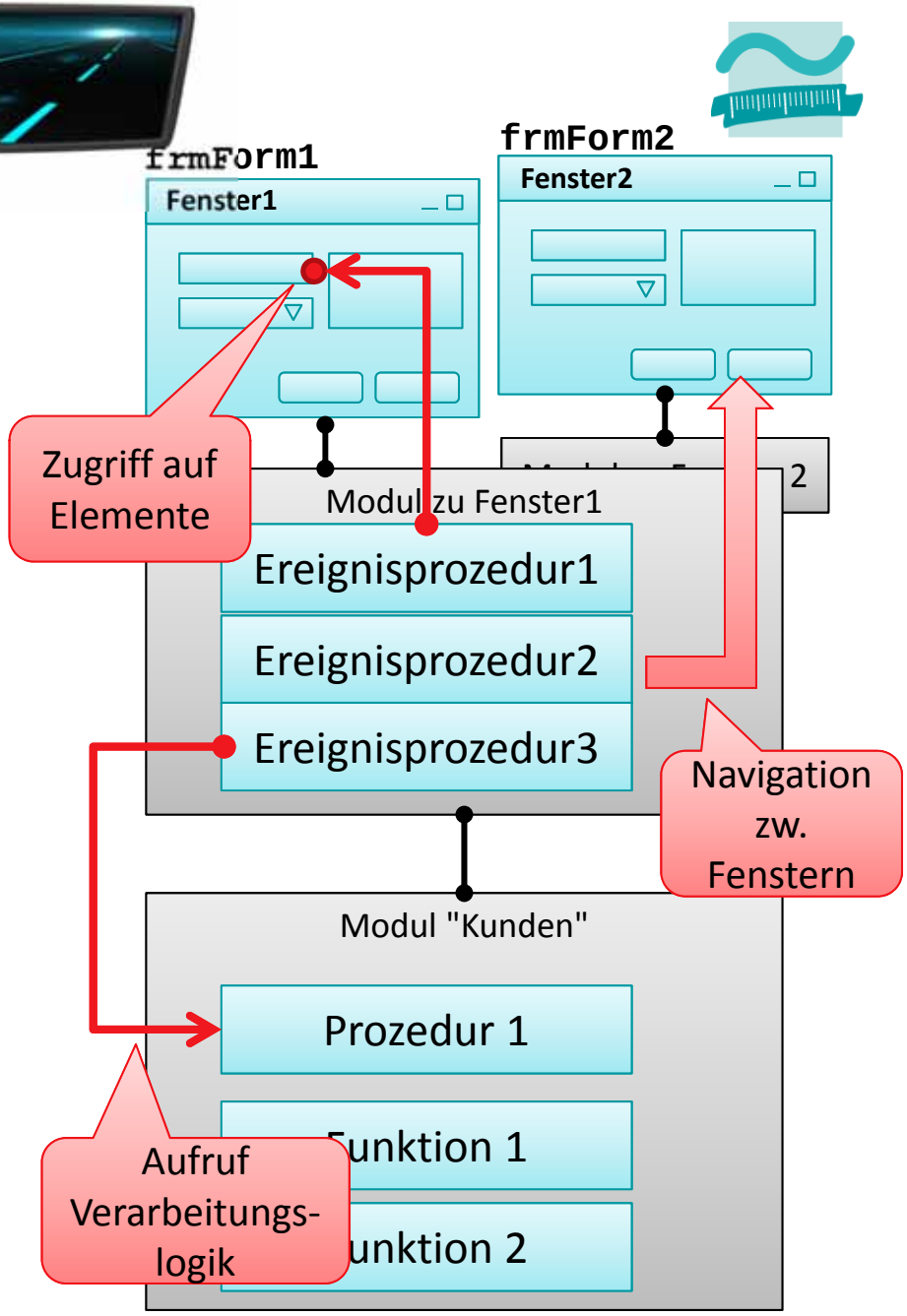
- Eingabe und Auswahl
  - Textfelder
  - Aufklappliste/Kombinationsfeld
  - Mehrfachauswahllisten
  - Radioknöpfe (Optionsfeld)
  - Kontrollkästchen (Checkbox)
- Aktionselemente
  - Schaltfläche (einfach)
  - Umschaltfläche (Troggle)
  - Menüeinträge
- Container
  - Rahmen/Gruppen
  - Registerkartensatz mit Registerkarten
  - Menüs
  - Fenster/Dialoge
    (in Access als Formulare)



Rückblick

Ereignisprozeduren bieten Zugriffsmöglichkeit auf die Oberflächenelemente und dienen zum

- Steuern der Elemente auf der Oberfläche
  - Aktivieren/Dekativieren bzw. Einblenden/Ausblenden von Elementen
  - Navigation zwischen Fenstern
  - ...
- Aufruf der Verarbeitungslogik
  - Übergabe der eingegebenen Daten zur Verarbeitung
  - Ermitteln der anzuzeigenden Daten
  - Ausführen von komplexen Berechnungen
  - ...



Rückblick

Referenzvariable Me stellt Funktionen zur Verfügung

- Zugriff auf den Wert von Feldern liefert immer String

```
' Generelle Syntax
Let <VarString> = Me.<BezeichnerDesFeldes>.Value
Let <VarZahl> = Val(Me.<BezeichnerDesFeldes>.Value)
```

- Genereller Zugriff auf Eigenschaften von Elementen

```
' Generelle Syntax
Let <Var> = Me.<Bez>.<Eigenschaft> ' Lesen
Let Me.<Bez>.<Eigenschaft> = <Var> ' Schreiben/Ändern
```

Beispiele

```
Let strName = Me.txtName.Value
Let intAlter = Val(Me.txtName.Value)

Let Me.txtName.Visible = False
```

Rückblick

Kommando DoCmd stellt Funktionen zur Verfügung

- Generelle Syntax zum Öffnen von Fenstern

```
' Generelle Syntax (Auszug)
DoCmd.OpenForm <Formularname>
```

- Generelle Syntax zum Schließen von Fenstern

```
' Generelle Syntax (Auszug)
DoCmd.Close <TypZuSchließendesObjekt>, <Name>
' Syntax zum Schließen von Formularen
DoCmd.Close acForm, <Formularname>
```

- Generelle Syntax zum Navigieren zwischen Fenstern

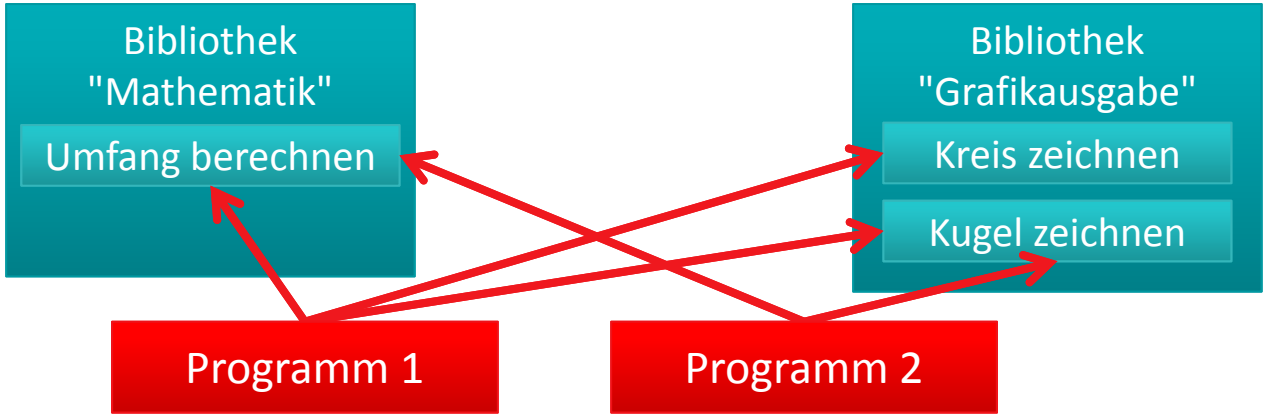
```
' Generelle Syntax (Auszug)
DoCmd.BrowseTo <TypZielOjekt>, <Name>
' Syntax zum Schließen von Formularen
DoCmd.BrowseTo acBrowseToForm, <Formularname>
```

Rückblick

Rückblick (LE02)

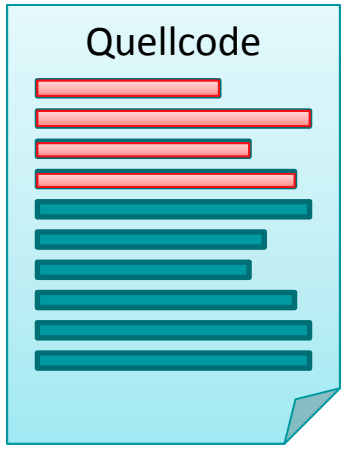
Bibliothek

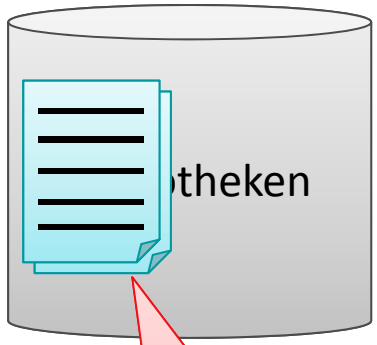
- Zusammenfassung von Programmteilen, die in anderen Programmen eingebunden und dadurch wiederverwendet werden können
- dient meist der Lösung einer abgegrenzten Funktionalität
- Beispiel



Rückblick (LE02)

Verwendung von Bibliotheken im Interpreter





Interpreter

Weitere Anweisungen werden entsprechend jeweils Anweisung für Anweisung verarbeitet.

Noch offen ist, wie Wir Bibliotheken in unseren Programmen verwenden können.

Quelle: [7]

Inhalt

Einordnung

Rückblick

Ausgangspunkt
- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem
- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl
- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff
- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Ausgangspunkt

Benutzer arbeitet über Benutzeroberfläche mit Anwendung

- optimiert für Endgeräte, z.B. Desktop, Web-Anwendung, Mobilgeräte

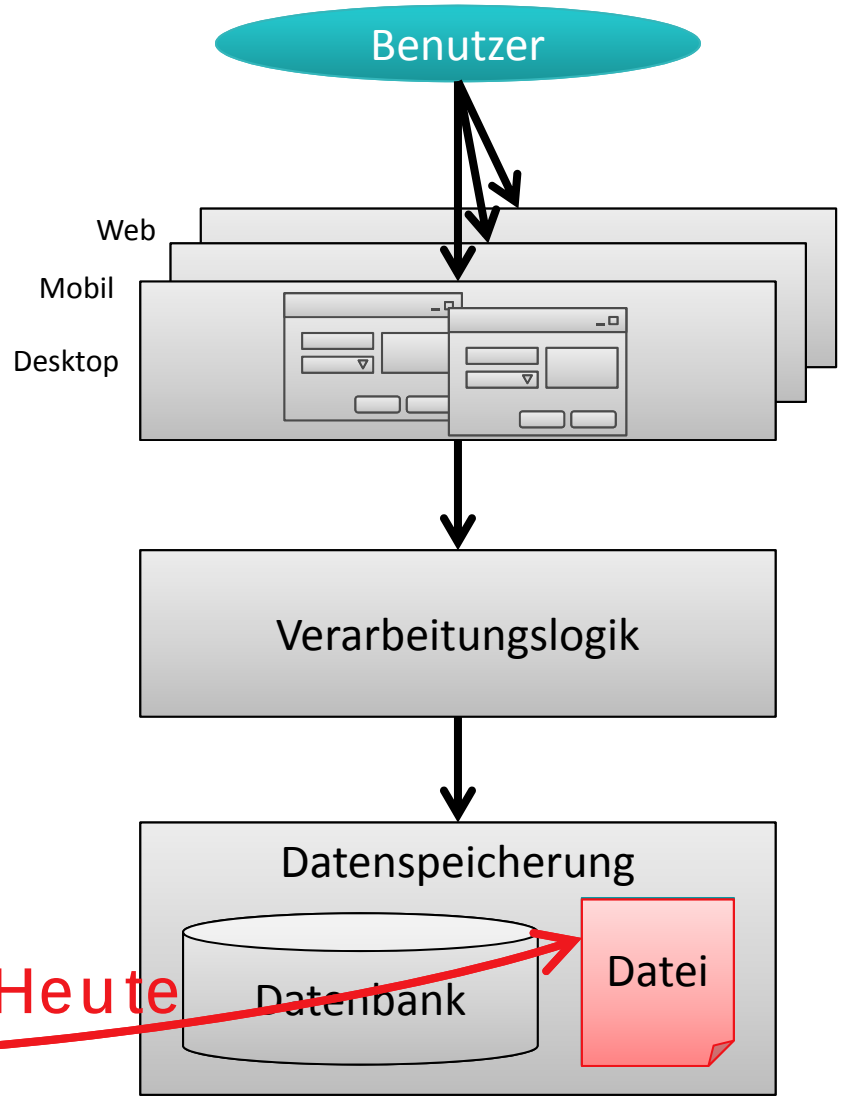
Benutzeroberfläche

- bietet Funktionen, zeigt Ausgabe und nimmt Eingabe entgegen
- nutzt Verarbeitungslogik außerhalb der Oberfläche

Verarbeitungslogik

- fachliche Algorithmen zur Verarbeitung der eingegebenen und Aufbereitung der auszugebenden Daten
- nutzt von der Datenspeicherung bereitgestellte Daten

Datenspeicherung bietet Zugriff auf die gespeicherten Daten (i.d.R. in einer Datenbank oder im Dateisystem gespeichert)



Dateisystem

Ablagestruktur für Daten auf Datenträgern

- Daten in der Regel als Dateien
- Abspeichern und Wiederauffinden erleichtern

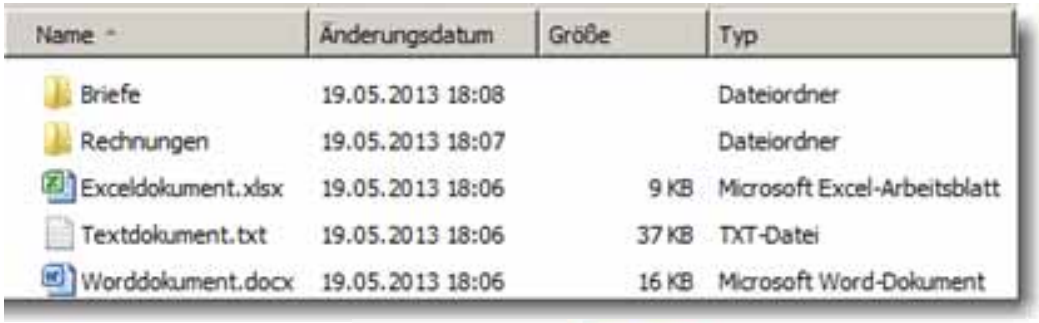
Zugriffsmöglichkeiten auf Daten in Dateien

- Lesen
- Schreiben/Ändern
- Löschen

Elemente im Dateisystem

Dateien

- Name inkl. Endung
- Weitere Eigenschaften, z.B.
  - Größe,
  - Änderungsdatum,
  - Schreibschutz
- sind in Verzeichnissen enthalten
- haben einen Pfad
  - absolut: ausgehend von Wurzel
  - relativ: ausgehend von anderem Verzeichnis

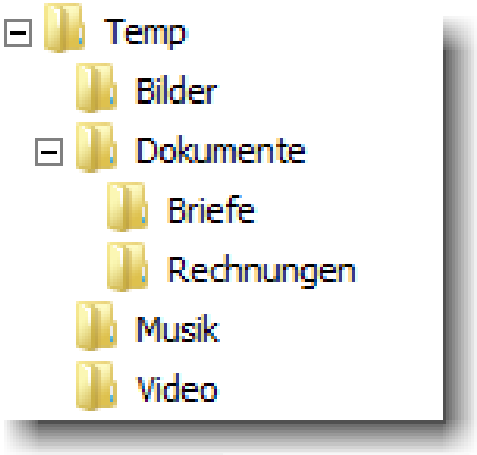


Name	Änderungsdatum	Größe	Typ
Briefe	19.05.2013 18:08		Dateiordner
Rechnungen	19.05.2013 18:07		Dateiordner
Exceldokument.xlsx	19.05.2013 18:06	9 KB	Microsoft Excel-Arbeitsblatt
Textdokument.txt	19.05.2013 18:06	37 KB	TXT-Datei
Worddokument.docx	19.05.2013 18:06	16 KB	Microsoft Word-Dokument

Elemente im Dateisystem

Verzeichnisse (Ordner)

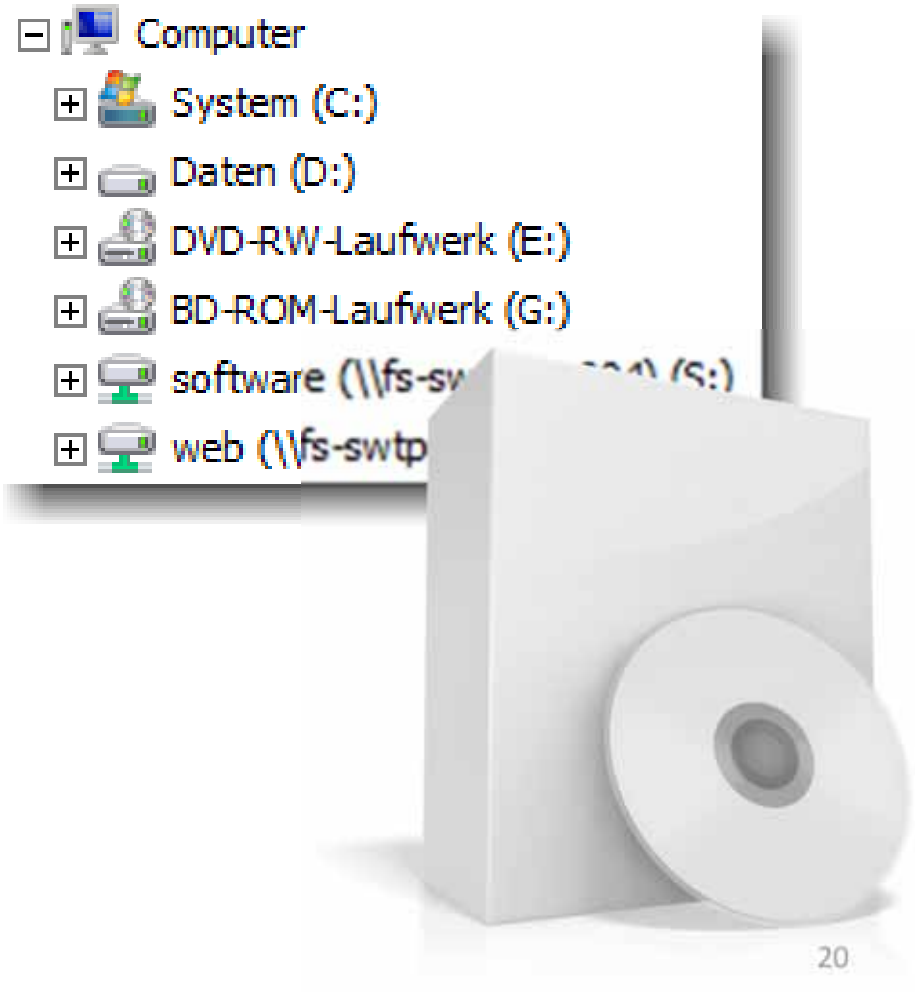
- haben einen Namen
- Eigenschaften
  - Versteckt
  - Größe (der enthaltenen Dateien)
  - Datum der Erstellung
  - …
- enthalten andere Verzeichnisse und/oder Dateien
- bilden Baumstruktur
- sind Teil eines Pfades



Elemente im Dateisystem

Laufwerke (unter Windows)

- haben Laufwerksbuchstaben und weitere Eigenschaften
  - Bezeichnung
  - Dateisystem
  - Speicherplatz
- enthalten Verzeichnisse und/oder Dateien



Elemente im Dateisystem

Weitere Elemente

- Links (symbolische Links, harte Links/Hardlinks)
- Dateien für spezielle Zwecke, z.B. zur Repräsentation von Geräten

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Zugriff auf Dateisystem

Auflisten von vorhandenen

- Laufwerken
- Verzeichnissen
- Dateien

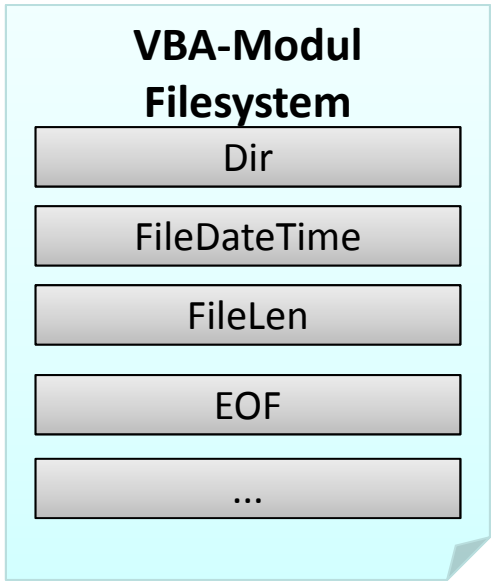
Verzeichnisse

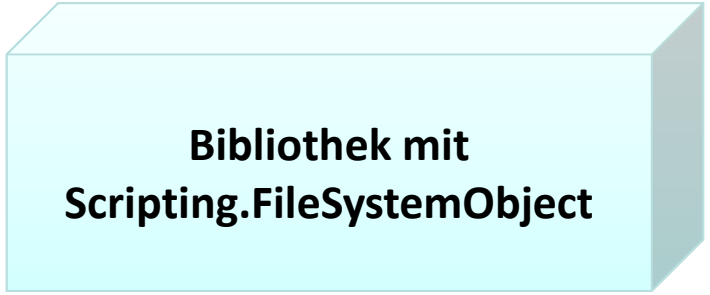
- Existenz prüfen
- Löschen
- Anlegen

Dateien

- Existenz prüfen
- Löschen
- Verschieben
- Kopieren

…





Modul "Filesystem"

Funktion Dir() zum Auflisten vorhandener Elemente

- Pfad zu einem Verzeichnis als Parameter
- liefert Namen des ersten enthaltenen Elementes
- bei jedem weiteren parameterlosem Aufruf liefert es Namen des nächsten Elementes oder leeren String, wenn am Ende

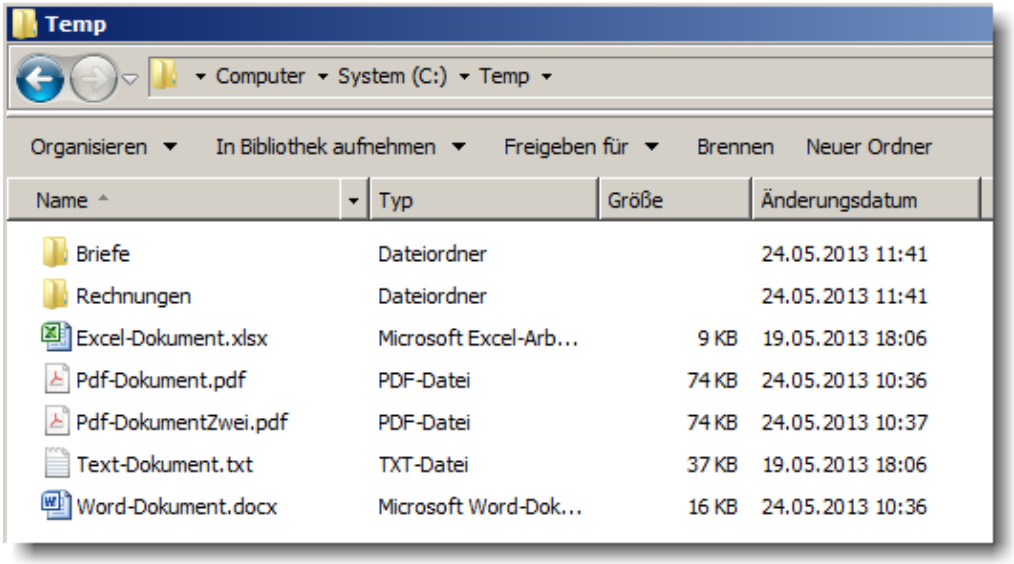
Syntax

```
' Generelle Syntax (Einfache Form)
Let <strElement> = Filesystem.Dir(<Pfad>) ' Erste Datei im Pfad
Let <strElement> = Filesystem.Dir() ' Nächste Datei im vorh. Pfad
```

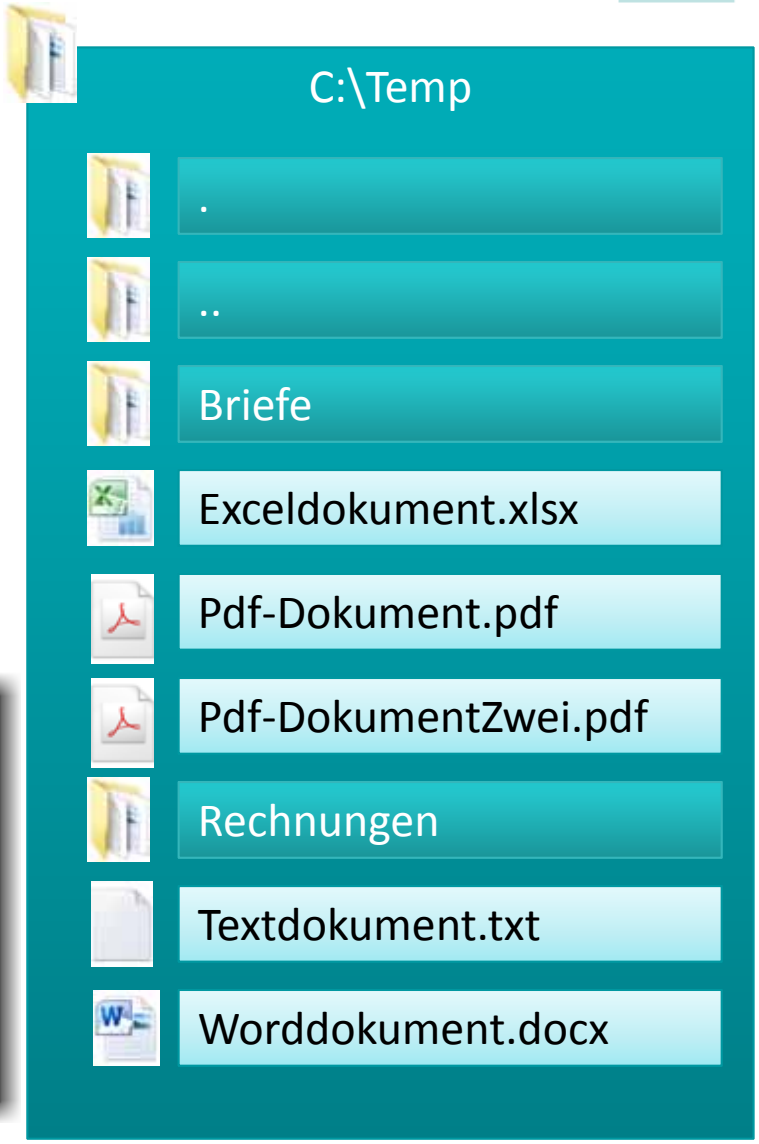
Beispiel

```
' Dir mit Platzhalter *.txt verwenden und erstes Element auslesen
Let strDateiname = FileSystem.Dir("C:\Temp\*.txt")
' Ausgabe des Namens der Datei
Debug.Print strDateiname
' Weiterschalten zur nächsten Datei
Let strDateiname = FileSystem.Dir()
```

Modul "FileSystem"




```
C:\>dir c:\temp
 Datenträger in Laufwerk C: ist System
 Volumeseriennummer: 4E26-F5E2

 Verzeichnis von c:\temp

24.05.2013  11:41    <DIR>          .
24.05.2013  11:41    <DIR>          ..
24.05.2013  11:41    <DIR>          Briefe
19.05.2013  18:06             8.833 Excel-Dokument.xlsx
24.05.2013  10:36            75.369 Pdf-Dokument.pdf
24.05.2013  10:37            75.475 Pdf-DokumentZwei.pdf
24.05.2013  11:41    <DIR>          Rechnungen
19.05.2013  18:06            37.120 Text-Dokument.txt
24.05.2013  10:36            16.330 Word-Dokument.docx
               5 Datei(en),        213.127 Bytes
               4 Verzeichnis(se), 314.862.747.648 Bytes frei
```



Modul "FileSystem"

Auflisten vorhandener PDF-Dateien

- Suche nach "*.pdf"
- im Verzeichnis C:\Temp
- entspricht Kommando
  `dir C:\Temp\*.pdf`

```
C:\>dir C:\Temp\*.pdf
 Datenträger in Laufwerk C: ist System
 Volumeseriennummer: 4E26-F5E2

 Verzeichnis von C:\Temp

24.05.2013  10:36            75.369 Pdf-Dokument.pdf
24.05.2013  10:37            75.475 Pdf-DokumentZwei.pdf
               2 Datei(en),        150.844 Bytes
               0 Verzeichnis(se), 315.010.101.248 Bytes frei
```

Modul "FileSystem"

Auflisten vorhandener PDF-Dateien

- Suche nach "*.pdf"
- im Verzeichnis C:\Temp

```
Let strName =
FileSystem.Dir("C:\Temp\*.pdf")
  Wenn strName <> "", dann ...
Let strName = FileSystem.Dir()
  Wenn strName <> "", dann ...
Let strName = FileSystem.Dir()
' Hier ist strName = ""
```

Modul "FileSystem": Beispiel 10.01

Ziel

- Auflisten von Inhalten des Verzeichnisses

Aufgabe

- Prozedur, die in einem gegebenen Verzeichnis
- die PDF-Dateien ermittelt und mit ihrem Namen
- im Direktbereich ausgibt

Modul "FileSystem": Beispiel 10.01

Lösung

```
' Variable für Dateinamen deklarieren
Dim strDateiname As String

' Dir mit Platzhalter *.* (für alle Dateien) verwenden
' und erstes Element auslesen
Let strDateiname = FileSystem.Dir("C:\Temp\*.pdf")

' Schleife solange, wie Dir() Dateinamen liefert
Do While strDateiname <> ""

 ' Ausgabe des Namens der Datei
 Debug.Print strDateiname
 ' Weiterschalten zur nächsten Datei
 Let strDateiname = FileSystem.Dir()

Loop
```

Modul "FileSystem"

Auflisten aller vorhandenen Dateien

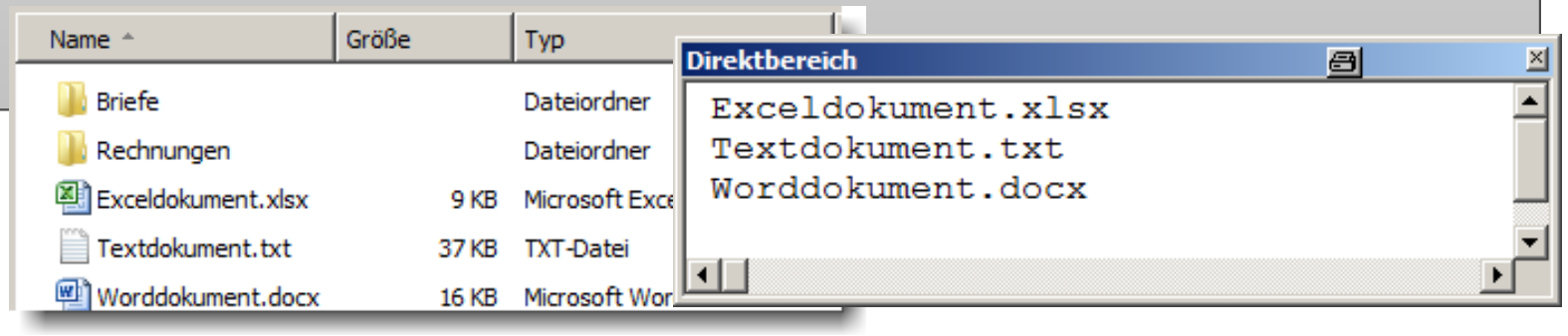
- Suche nach "*" bzw. "*.*"
- im Verzeichnis C:\Temp

```
Let strName =
FileSystem.Dir("C:\Temp\*.*")
  Wenn strName <> "", dann ...
Let strName = FileSystem.Dir()
  Wenn strName <> "", dann ...
Let strName = FileSystem.Dir()


  Wenn strName <> "", dann ...
Let strName = FileSystem.Dir()
  Wenn strName <> "", dann ...
Let strName = FileSystem.Dir()
  Wenn strName <> "", dann ...
Let strName = FileSystem.Dir()
' Hier ist strName = ""
```

Modul "FileSystem"

Beispiel mit der Funktion Dir()

```
' Variable für Dateinamen deklarieren
Dim strDateiname As String

' Dir mit Platzhalter *.* (für alle Dateien) verwenden
' und erstes Element auslesen
Let strDateiname = FileSystem.Dir("C:\Temp\*.*")

' Schleife solange, wie Dir() Dateinamen liefert
Do While strDateiname <> ""

 ' Ausgabe des Namens der Datei
 Debug.Print strDateiname
 ' Weiterschalten zur nächsten Datei
 Let strDateiname = FileSystem.Dir()

Loop
```



Modul "Filesystem"

Funktion Dir() zum Auflisten vorhandener Elemente

- Pfad zu einem Verzeichnis als Parameter und per Parameter Steuerung ob zusätzlich auch Systemdateien, Verzeichnisse usw. gefunden werden sollen
- bei jedem weiteren parameterlosem Aufruf liefert es Namen des nächsten Elementes oder leeren String, wenn am Ende

Syntax

```
' Generelle Syntax mit Angabe des gewünschten Inhalts
' z.B. vbDirectory, vbHidden, vbSystem
Let <strElement> = Dir(<Pfad>, <ZusätzlicheGewünschterInhalte>)
Let <strElement> = Dir() ' Nächste Datei (im vorherigen Pfad)
```

Beispiel

```
' Dir verwenden und auch Verzeichnisse auflisten
Let strDateiname = FileSystem.Dir("C:\Temp", vbDirectory)
```

Modul "FileSystem"

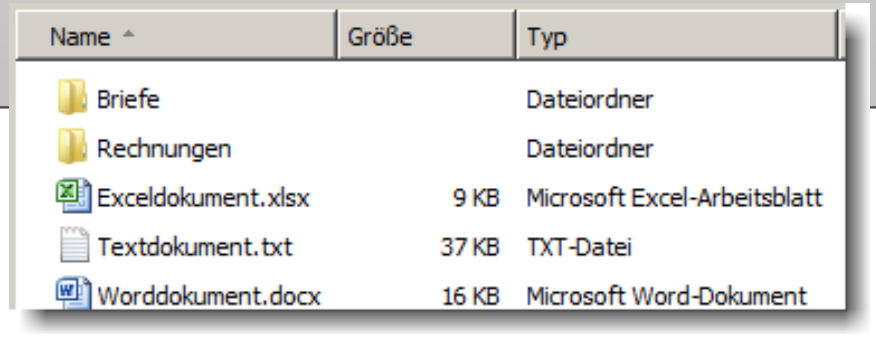
Alles auflisten (auch Verz.)

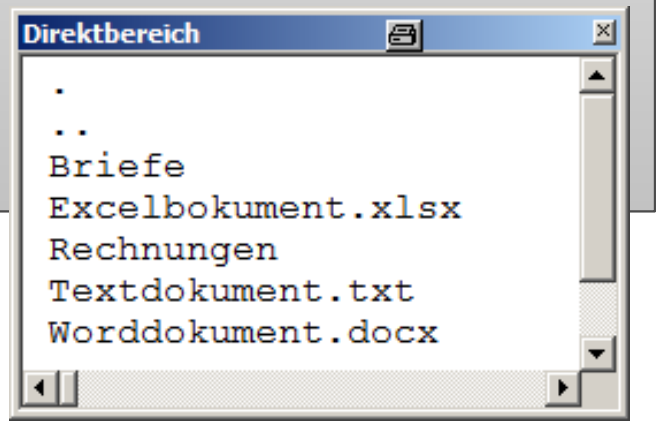
```
Let strName =
FileSystem.Dir("C:\Temp\*.*", vbDirectory)

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()

          Wenn strName <> "", dann ...
    Let strName = FileSystem.Dir()
    ' Hier ist strName = ""
```

Modul "FileSystem"

Beispiel mit der Funktion Dir()

```
' Variable für Dateinamen deklarieren
Dim strDateiname As String

' Dir mit Platzhalter *.* (für alle Dateien) verwenden
' und erstes Element auslesen
Let strDateiname = FileSystem.Dir("C:\Temp\*.*", vbDirectory)

' Schleife solange, wie Dir() Dateinamen liefert
Do While strDateiname <> ""

 ' Ausgabe des Namens der Datei
 Debug.Print strDateiname
 ' Weiterschalten zur nächsten Datei
 Let strDateiname = FileSystem.Dir()

Loop
```





Modul "FileSystem"

Weitere Prozeduren/Funktionen

– Verzeichnis anlegen (Make Directory)

```
' Generelle Syntax
Call Filesystem.MkDir(<Pfad>)
```

```
' Beispiele
' C:\Temp muss bereits existieren
Call FileSystem.MkDir("C:\Temp\Neu")
' Jetzt kann auch GanzNeu angelegt werden
Call FileSystem.MkDir("C:\Temp\Neu\GanzNeu")
```

– leeres Verzeichnis löschen (Remove Directory)

```
' Generelle Syntax
Call Filesystem.RmDir(<Pfad>) ' Verzeichnis muss leer sein
```

```
' Beispiele
' Erst GanzNeu löschen
Call FileSystem.RmDir("C:\Temp\Neu\GanzNeu")
' Jetzt kann auch Neu gelöscht werden
Call FileSystem.RmDir("C:\Temp\Neu")
```

Modul "FileSystem"

Weitere Prozeduren/Funktionen

– Datei kopieren

```
' Generelle Syntax
Call Filesystem.FileCopy(<QuellePfadDatei>, <ZielPfadDatei>)
```

```
' Beispiele
Call FileSystem.FileCopy("C:\Temp\doc1.txt", "C:\Temp\doc2.txt")
```

– Änderungsdatum ermitteln

```
' Generelle Syntax
Let <strVariable> = Filesystem.FileDateTime(<PfadDatei>)
```

```
' Beispiele
Let strZeitpnkt = Filesystem.FileDateTime("C:\Temp\doc1.txt")
Debug.Print strZeitpunkt
Debug.Print Filesystem.FileDateTime("C:\Temp\doc2.txt")
```

Zugriff auf Dateisystem

Auflisten von vorhandenen

- Laufwerken
- Verzeichnissen
- Dateien

Verzeichnisse

- Existenz prüfen
- Löschen
- Anlegen

Dateien

- Existenz prüfen
- Löschen
- Verschieben
- Kopieren

…

VBA-Modul Filesystem

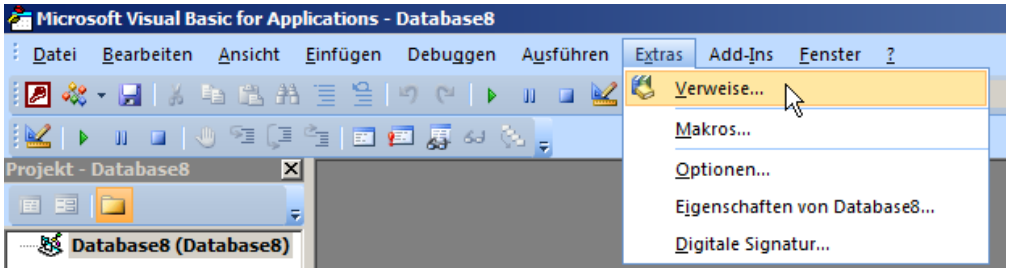
- Dir
- FileDateTime
- FileLen
- EOF
- …

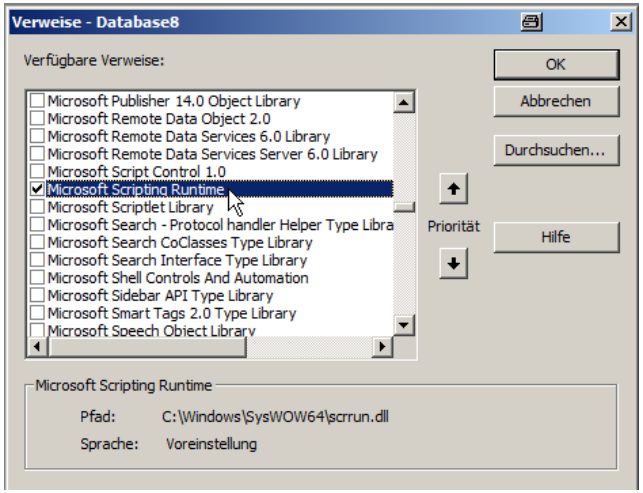
Bibliothek mit Scripting.FileSystemObject

FileSystem-Objekt aus MS Scripting Runtime

Erweiterte Möglichkeiten durch Zugriff auf Filesystem-Objekt

- Bereitgestellt durch Bibliothek "Microsoft Scripting Runtime"
- Bibliothek muss eingebunden werden, um ihre Funktionen nutzen zu können
  - Im VBA-Editor > Extras > Verweise
  - Im Dialog "Verweise" den Eintrag "Microsoft Scripting Runtime" aktivieren > OK





FileSystem-Objekt aus MS Scripting Runtime

FileSystem deklarieren und erzeugen

```
' Deklaration und Initialisierung
 Dim <FileSysObj> As FileSystemObject
 Set <FileSysObj> = New FileSystemObject
```

Komfortable Funktionen

– für Kopieren, Verschieben und Löschen von Verzeichnissen und Dateien

```
' Beispiele
 Dim oFs As FileSystemObject
 Set oFs = New FileSystemObject

 Call oFs.CopyFile("C:\Temp\doc1.txt", "C:\Temp\doc2.txt")
 Call oFs.DeleteFile("C:\Temp\doc1.txt")
 Call oFs.MoveFile("C:\Temp\doc2.txt", "C:\Temp\doc1.txt")
```

– ...

FileSystem-Objekt aus MS Scripting Runtime

Komfortable Funktionen (Fortsetzung)

- für Zugriff auf Laufwerke (**`GetDrive()`**, **`GetDriveName()`**)
- für Dateinamen und Erweiterungen (**`GetFileName()`**, **`GetExtensionName()`**)
- für Prüfung auf Vorhandensein (**`DriveExists()`**, **`FileExists()`** und **`FolderExists()`**)
- Details unter: http://msdn.microsoft.com/en-us/library/aa242706%28v=vs.60%29.aspx

```
' Beispiele
 Dim oFs As FileSystemObject
 Set oFs = New FileSystemObject

 Call oFs.CopyFile("C:\Temp\doc1.txt", "C:\Temp\doc2.txt")
 If oFs.FileExists("C:\Temp\doc2.txt") Then
  MsgBox ("Datei ist da!")
 Else
  MsgBox ("Datei ist nicht da!")
 End If
```

FileSystem-Objekt: Beispiel 10.02

Ziel

- Einbindung der Bibliothek "Microsoft Scripting Runtime"
- Verwendung des FileSystem-Objekts zum Auslesen von Laufwerken

Aufgabe: Schreiben Sie eine Prozedur,

- die ein FileSystem-Objekt erzeugt und mit dessen Hilfe
- Informationen über das Laufwerk C des Computers im Direktbereich ausgibt
  - Größe insgesamt
  - Freier Speicherplatz
  - Dateisystem
  - …

FileSystem-Objekt: Beispiel 10.02

Lösung

```
' Variablen deklarieren und initialisieren
 Dim fs As FileSystemObject
 Set fs = New FileSystemObject

' Zugriff auf Laufwerk C in Collectio Drives
 With fs.Drives.Item("C")
  Debug.Print "Laufwerk " & .DriveLetter & ":"
  Debug.Print "Dateisystem: " & .FileSystem
  Debug.Print "Größe insgesamt: " & _
                Round(.TotalSize / (1024 ^ 3), 0) & " GB"
  Debug.Print "Freier Speicher: " & _
                Round(.FreeSpace / (1024 ^ 3), 0) & " GB"
 End With
```

Zugriff auf Dateisystem

Auflisten von vorhandenen

- Laufwerken
- Verzeichnissen
- Dateien

Verzeichnisse

- Existenz prüfen
- Löschen
- Anlegen

Dateien

- Existenz prüfen
- Löschen
- Verschieben
- Kopieren

VBA-Modul Filesystem

- Dir
- FileDateTime
- FileLen
- EOF
- ...

Objekt Scripting.FileSystemObject

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Dialoge zur Datei- und Verzeichnisauswahl

Einsatz

- sinnvoll, wenn vom Benutzer das Ziel zum Speichern oder Laden von Daten im Dateisystem selbst gewählt werden soll
- als Dateiauswahl
  - zum Lesen einer oder mehrerer Dateien
  - zum Speichern einer Datei
- als Verzeichnisauswahl
  - zum Lesen des gesamten Verzeichnisinhalts
  - zum Speichern mehrerer Dateien mit fest vorgegebenem Namen

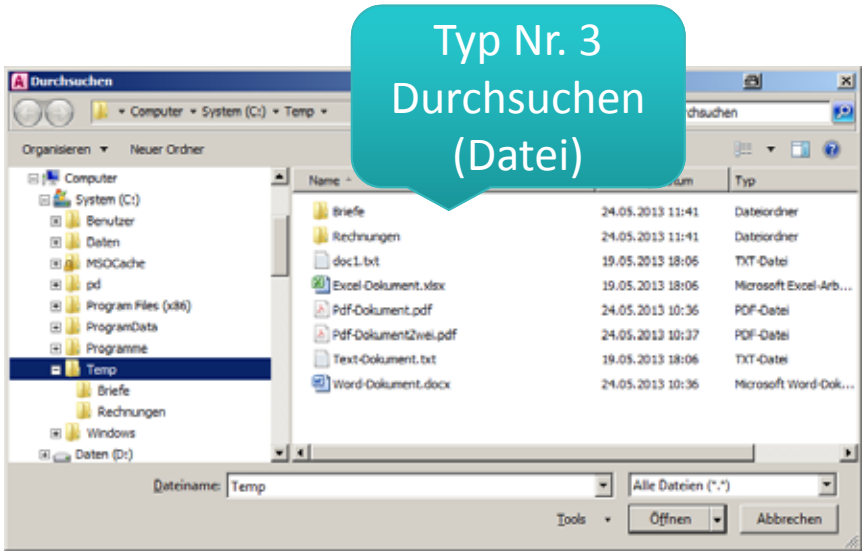
Standarddialoge zur Datei-/Verzeichnisauswahl

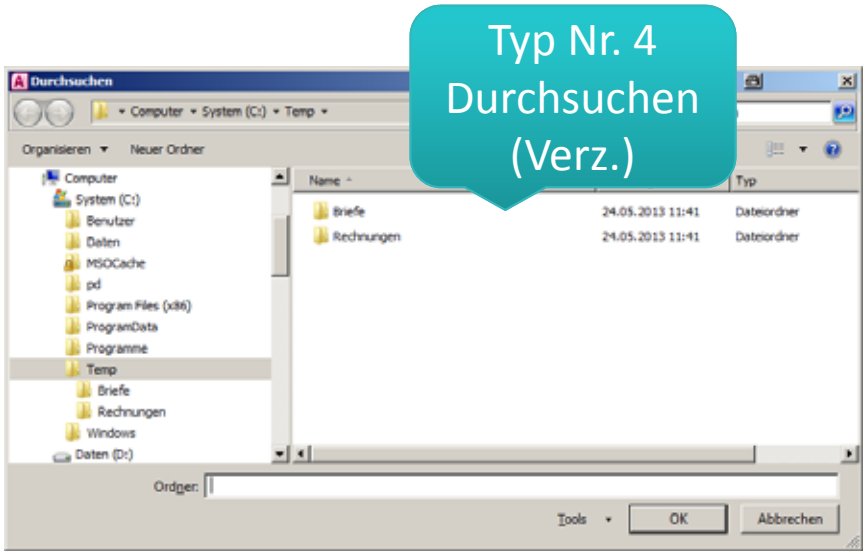
Standardmäßig in MS Access ab Version 2010

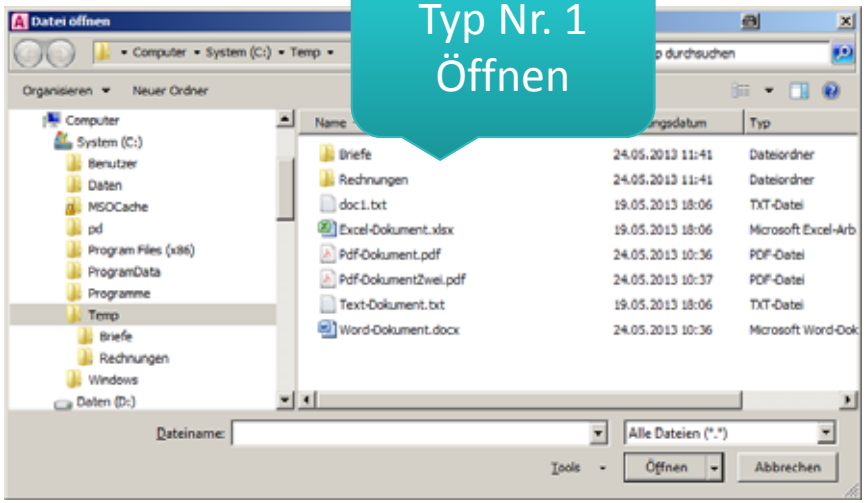
- unpraktische Möglichkeiten Dialoge zu verwenden für
  - Auswahl einer Datei
  - Auswahl mehrerer Dateien
  - Auswahl eines Verzeichnisses
  - Wahl einer Datei zum Speichern
- Typ des Dialogs wird durch Zahlenwert repräsentiert

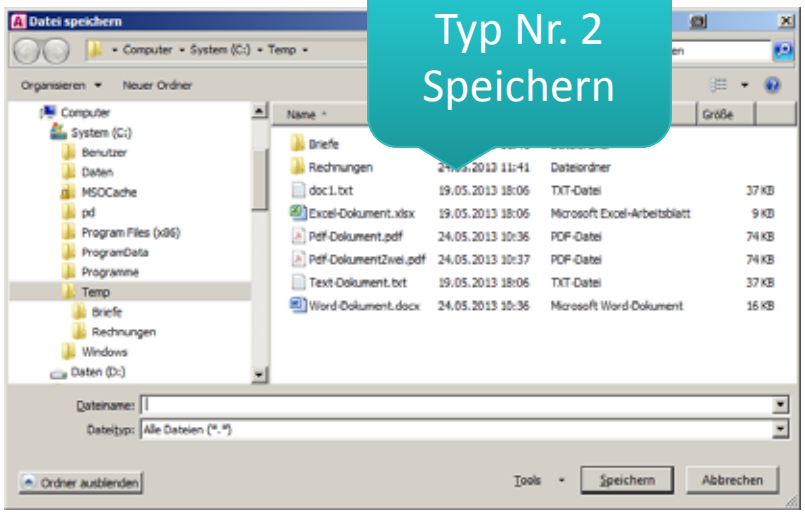
Dialogtyp	Zahlenwert
Dateiauswahl/-mehrfachauswahl (Durchsuchen)	3
Verzeichnisauswahl (Durchsuchen)	4
Datei öffnen	1
Datei speichern unter	2

Standarddialoge zur Datei-/Verzeichnisauswahl









Standarddialoge zur Datei-/Verzeichnisauswahl

Generelle Syntax

– Deklaration und Initialisierung

```
Dim <FileDlgObj> As Object
Set <FileDlgObj> = Application.FileDialog(<Zahl>)
```

– Konfiguration (z.B. Mehrfachauswahl)

```
<FileDlgObj>.AllowMultiSelect = True
```

– Anzeige

```
Let <intVar> = <FileDlgObj>.Show() ' Rückgabewert 0 = Abbruch
```

– Ergebnis in Collection "SelectedItems" enthalten

```
<FileDlgObj>.SelectedItems
```

Standarddialoge zur Datei-/Verzeichnisauswahl

Beispiel (Einfachauswahl)

```
Sub beispielEinfachauswahl()

 Dim intResult As Integer ' Rückgabewert, ob Benutzer Dialog abbricht
 Dim oFd As Object ' Variable für FileDialog
 Set oFd = Application.FileDialog(1) ' Init. als Öffnen-Dialog

 oFd.AllowMultiSelect = False ' Konfiguration, z.B. Einfachauswahl

 Let intResult = oFd.Show ' anzeigen, merken welche Schaltfläche

 If intResult = 0 Then
  Exit Sub ' Abbruch durch Benutzer, Prozedur vorzeitig verlassen
 End If

 ' Prüfen, ob Datei gewählt wurde
 If oFd.SelectedItems.Count > 0 Then
  Debug.Print oFd.SelectedItems(1) ' Ausgabe der ausgewählten Datei
 Else
  Debug.Print "Keine Auswahl."
 End If

End Sub
```

Standarddialoge: Beispiel 10.03

Ziel

- Standarddialoge verwenden und ungeschickte Umsetzung erkennen

Aufgabe

- Implementieren Sie mit einem Standarddialog eine Mehrfachauswahl von Dateien
- Nachdem der Benutzer im Dialog mindestens eine Datei ausgewählt hat, geben Sie die ausgewählten Dateien im Direktbereich aus

Standarddialoge: Demo 10.03

Lösung

```
Sub beispielMehrfachauswahl()

 Dim intResult As Integer ' Rückgabewert, ob Benutzer Dialog abbricht
 Dim i As Integer ' Schleifenvariable
 Dim oFd As Object ' Variable für FileDialog
 Set oFd = Application.FileDialog(3) ' Initialisierung als Typ 3

 oFd.AllowMultiSelect = True ' Konfiguration, z.B. Mehrfachauswahl

 Let intResult = oFd.Show ' anzeigen, merken welche Schaltfläche

 If intResult = 0 Then
  Exit Sub ' Abbruch durch Benutzer, Prozedur vorzeitig verlassen
 End If

 ' Schleife über alle ausgewählten Dateien
 For i = 1 To oFd.SelectedItems.Count
  Debug.Print oFd.SelectedItems(i) ' Ausgabe der gewählten Dateien
 Next

End Sub
```

Standarddialoge zur Datei-/Verzeichnisauswahl

Ungeschickte Umsetzung der Standarddialoge

- Nachteile
  - Zahlenwerte anstelle sonst üblicher Verwendung von Konstanten
  - Keine Eingabeunterstützung auf dem FileDialog-Objekt
- Vorteil: Nutzbar ohne zusätzliche Bibliotheken
- Ursache: Late Binding

Hinweis: In früheren Versionen von MS Access

- mit Application.GetOpenFilename Dialog geöffnet
- anhand des Rückgabewertes entschieden, ob Abbruch oder Dateiauswahl
- wird in MS Access 2010 nicht (mehr) unterstützt

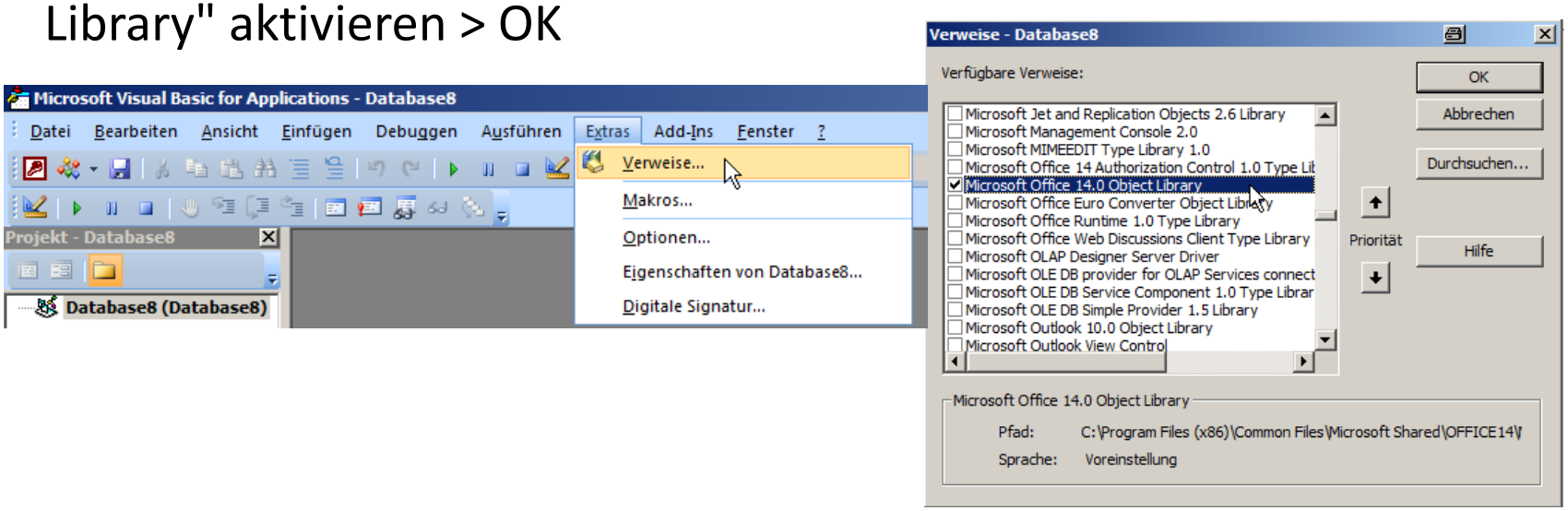
Inhalt

Inhalt

FileDialog-Objekt der MS Office Object Library

Erweiterte Möglichkeit durch Nutzung des FileDialog

- Nutzung der Bibliothek "Microsoft Office 14.0 Object Library" bietet erweiterte Dialoge für die Dateiauswahl
- Bibliothek muss eingebunden werden, um ihre Funktionen nutzen zu können
  - Im VBA-Editor > Extras > Verweise
  - Im Dialog "Verweise" den Eintrag " Microsoft Office 14.0 Object Library" aktivieren > OK

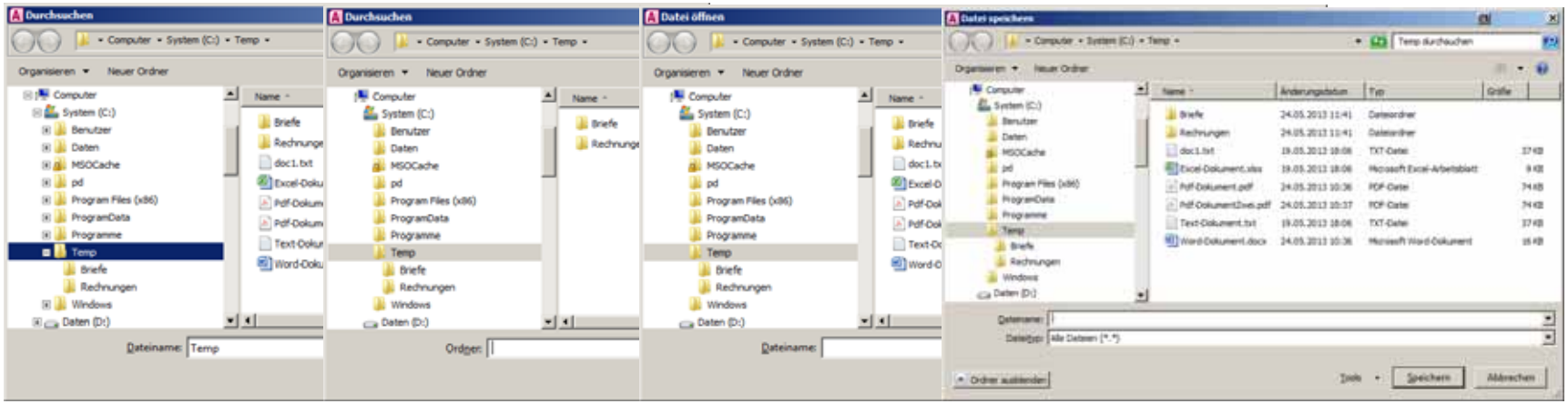


FileDialog-Objekt der MS Office Object Library

Deklaration und Erzeugung

```
' Generelle Syntax zur Deklaration und Initialisierung
Dim <FileDlgObj> As FileDialog
Set <FileDlgObj> = Application.FileDialog(<DialogTyp>)
```

Typ	Konstante	Wert
Dateiauswahl/-mehrfachauswahl	`msoFileDialogFilePicker`	3
Verzeichnisauswahl	`msoFileDialogFolderPicker`	4
Datei öffnen	`msoFileDialogOpen`	1
Datei speichern unter	`msoFileDialogSaveAs`	2

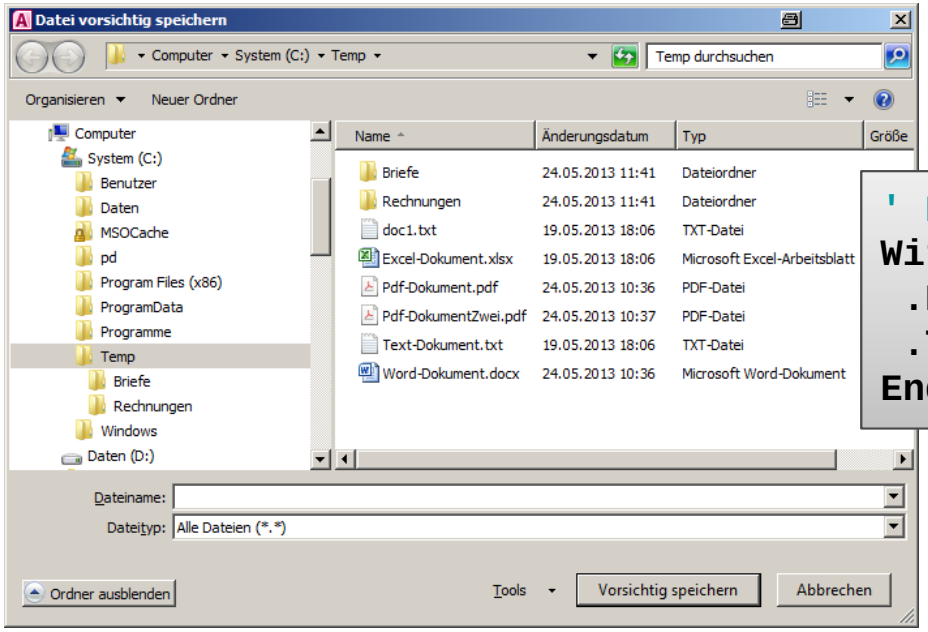


FileDialog-Objekt der MS Office Object Library

Konfiguration

– Objekt ermöglicht die Konfiguration verschiedener Eigenschaften (z.B. Schaltflächenbezeichnung, Dialogtitel)

```
' Konfiguration (Auszug)
<FileDlgObjekt>.ButtonName = <StringAusdruckWert>
<FileDlgObjekt>.Title = <StringAusdruckWert>
```




```
' Beispiel (Auszug)
With oFileDlg
  .ButtonName = "Jetzt speichern"
  .Title = "Vorsichtig speichern"
End With
```

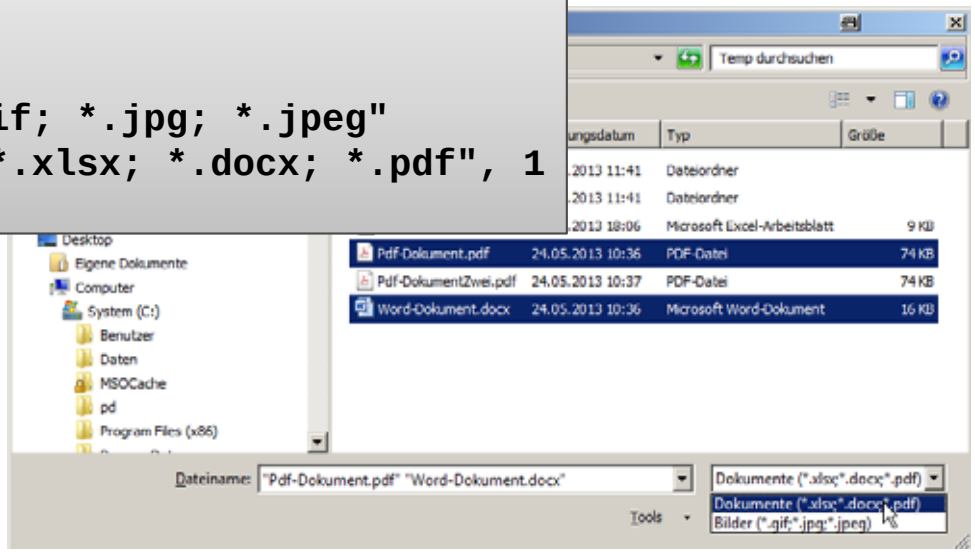
FileDialog-Objekt der MS Office Object Library

Konfiguration (Fortsetzung)

- Abhängig vom Dialogtyp weitere Eigenschaften möglich
  - Mehrfachauswahl von Dateien

```
' Konfiguration (Auszug)
<FileDlgObjekt>.Filters.Clear ' Vorhandene löschen
<FileDlgObjekt>.Filters.Add (<Name>, <Filter>, <Reihenflg>)
<FileDlgObjekt>.AllowMultiSelect = True|False
```

```
' Beispiel (Auszug)
With oFileDlg
 .AllowMultiSelect = True
 .Filters.Clear
 .Filters.Add "Bilder", "*.gif; *.jpg; *.jpeg"
 .Filters.Add "Dokumente", "*.xlsx; *.docx; *.pdf", 1
End With
```



FileDialog-Objekt der MS Office Object Library

Anzeige des Dialogs

– FileDialog wird mit Show() angezeigt

```
' Anzeigen
Let <intVar> = <FileDlgObjekt>.Show()
```

– Rückgabewert gibt Auskunft, ob Abgebrochen

- Wert -1 = Standardschaltfläche betätigt
- Wert 0 = Dialog abgebrochen

– Beispiel (Auszug)

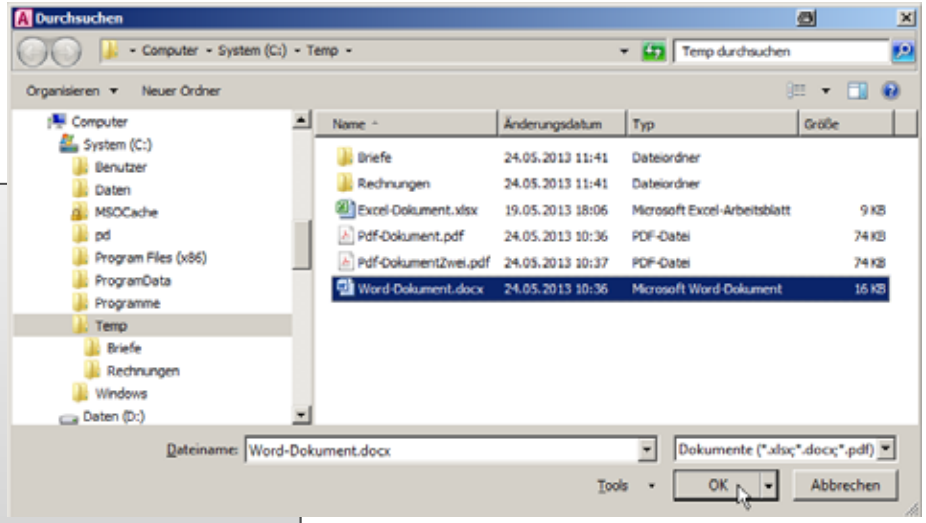
```
' Anzeigen
Let intResult = oFileDlg.Show()

' Ergebnis prüfen
If intResult = -1 Then
 Call MsgBox("Auswahl erfolgt")
Else
 Call MsgBox("Auswahl abgebrochen")
End If
```

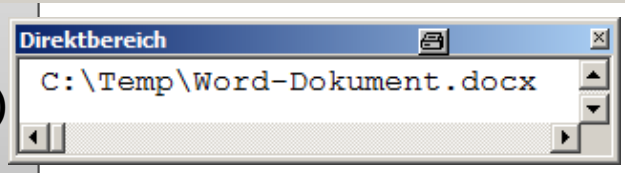
FileDialog-Objekt der MS Office Object Library

Auswertung der Ergebnisse (Einfachauswahl)

- Ergebnis der Auswahl ist in Collection SelectedItems enthalten
- Einfachauswahl
  - Prüfung, ob mind. ein Element ausgewählt wurden,
  - dann Zugriff auf Element
- Beispiel (Auszug)

```
' Einfachauswahl
oFileDlg.AllowMultiSelect = False

' Anzeigen
Let intResult = oFileDlg.Show()

' Ergebnis prüfen und ausgeben
If intResult = -1 Then
 If oFileDlg.SelectedItems.Count > 0 Then
    Debug.Print oFileDlg.SelectedItems.Item(1)
 End If
End If
```



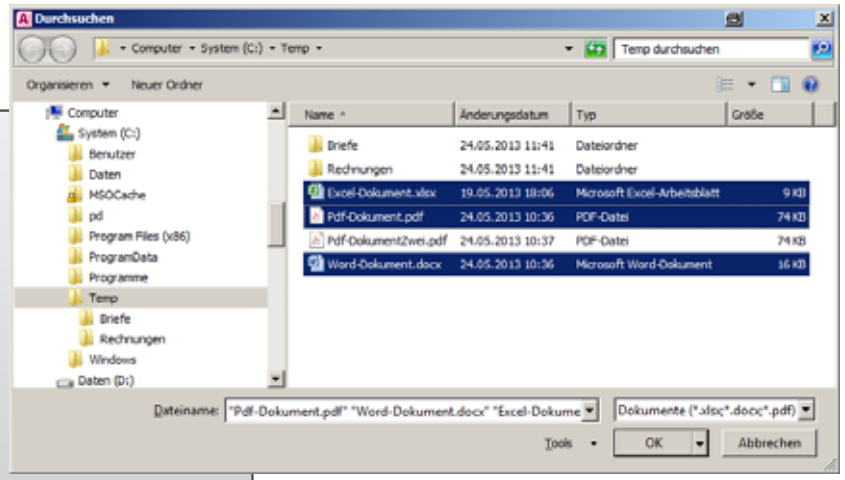


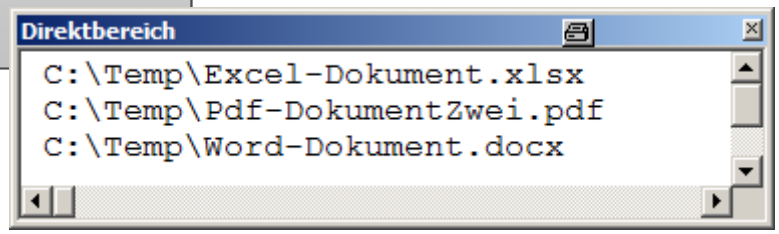
FileDialog-Objekt der MS Office Object Library

Auswertung der Ergebnisse (Mehrfachauswahl)

- Mehrfachauswahl: Schleife über Elemente der Collection
- Beispiel (Auszug)

```
' Einfachauswahl
oFileDlg.AllowMultiSelect = True

' Anzeigen
Let intResult = oFileDlg.Show()

' Ergebnis prüfen und ausgeben
If intResult = -1 Then
 For i = 1 To oFileDlg.SelectedItems.Count
    Debug.Print oFileDlg.SelectedItems.Item(i)
 Next
End If
```





FileDialog-Objekt der MS Office Object Library

Weitere Details

- http://msdn.microsoft.com/en-us/library/office/ff862446.aspx

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

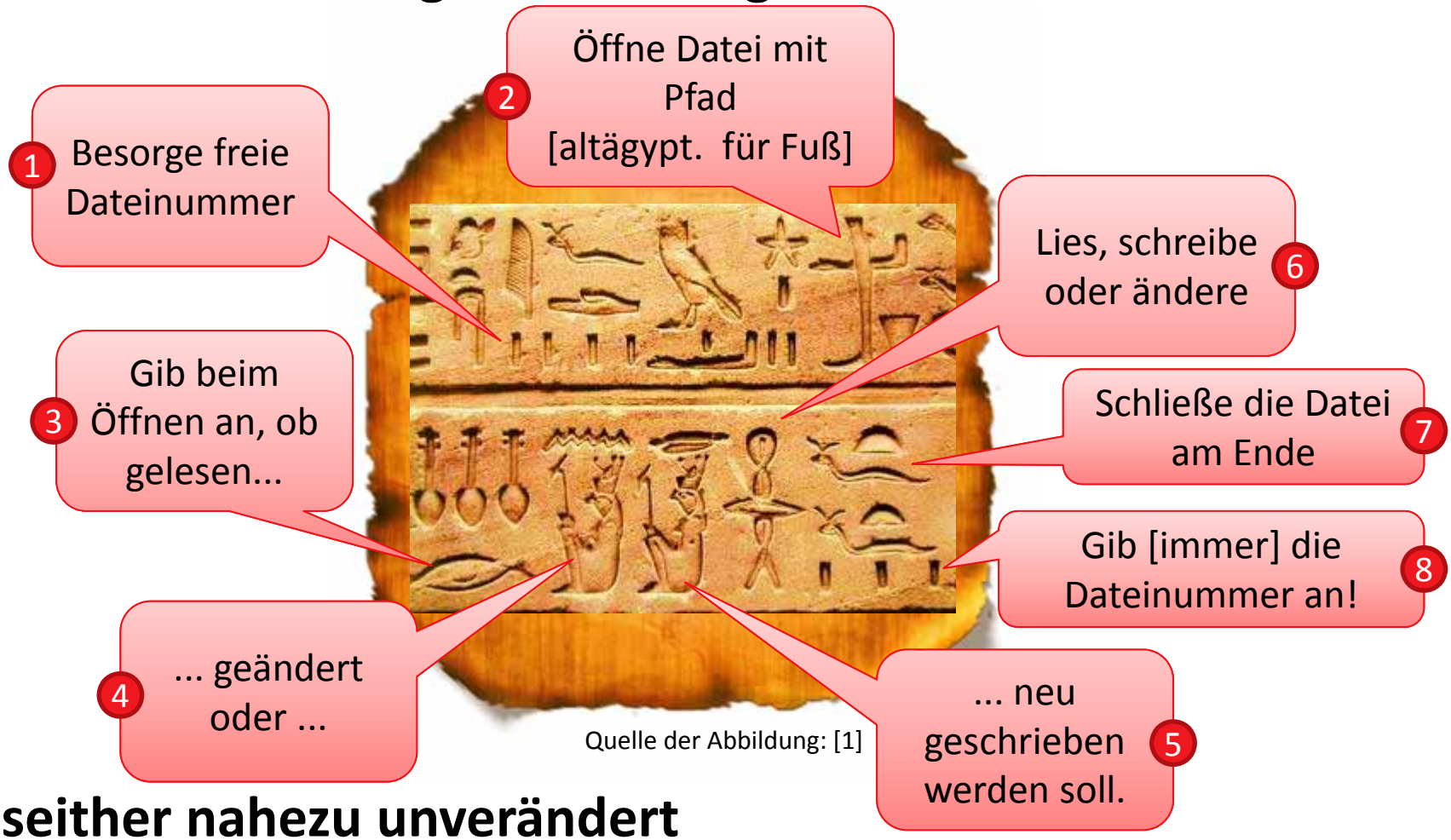
- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Grundlagen

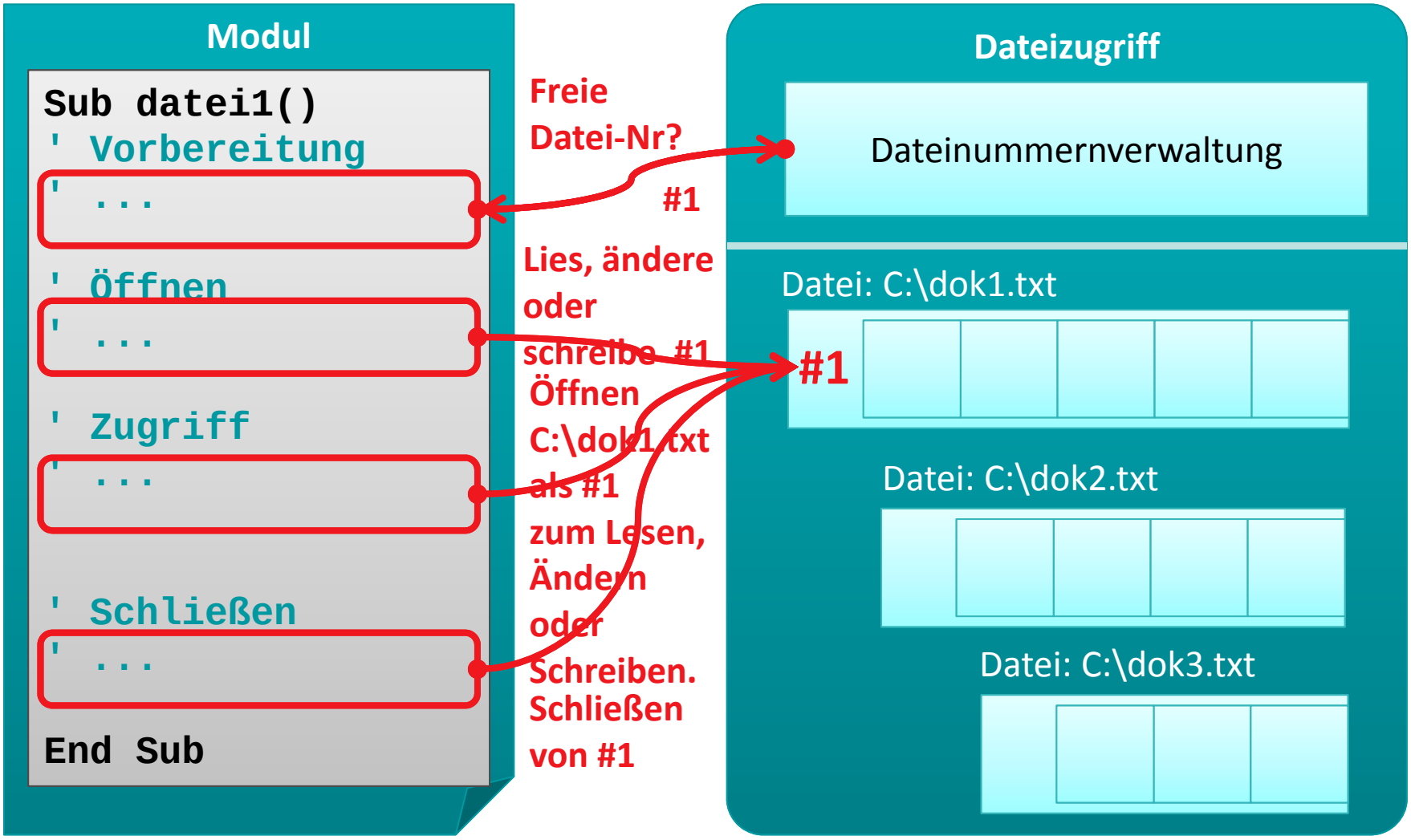
Früheste Darstellung des Dateizugriffs in Basic



Quelle der Abbildung: [1]

... seither nahezu unverändert

Grundlagen


Modul
Sub datei1()
' Vorbereitung
' ...
' Öffnen
' ...
' Zugriff
' ...
' Schließen
' ...
End Sub
Freie Datei-Nr?
#1
Lies, ändere oder schreibe #1
Öffnen C:\dok1.txt als #1
zum Lesen, Ändern oder Schreiben.
Schließen von #1
Dateizugriff
Dateinummernverwaltung
Datei: C:\dok1.txt
#1
Datei: C:\dok2.txt
Datei: C:\dok3.txt

Zugriff auf Dateien

Generelle Syntax

– Freie Dateinummer ermitteln

```
Let <intVar> = FileSystem.FreeFile()
```

– Öffnen einer Datei (verkürzte Form)

```
Open <Pfad> For <Modus> Access <Zugriff> As #<DateiNr>
```

- Pfad: Angabe des Pfades zur Datei
- Modus: Lesen (Input), Schreiben (Output), Ändern (Append), ...
- Zugriff: Lesen (Read), Schreiben (Write) oder Ändern (ReadWrite)
- Dateinummer: Zuvor mit FreeFile() ermittelte Nummer

– Schließen einer Datei

```
Close #<DateiNr>
```

Zugriff auf Dateien

Generelle Syntax

– Datei schreiben (Write)

```
Write #<DateiNr>, <WertAusdr> ' Variante 1
Write #<DateiNr>, <WertAusdr1>, <WertAusdr2>, ... ' Variante 2
Write #<DateiNr>, ' Leere Zeile
```

– Datei lesen (Input)

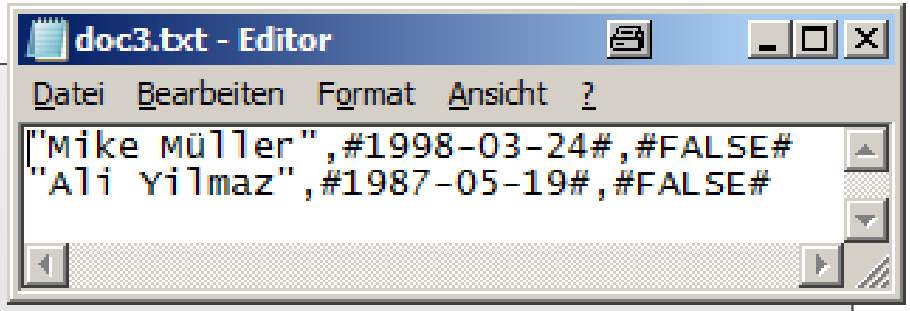
```
Input #<DateiNr>, <Variable> ' Variante 1
Input #<DateiNr>, <Var1>, <Var2>, ... ' Variante 2
```

– Datei zeilenweise lesen (Line Input)

```
Line Input #<DateiNr>, <StringVariable>
```

Zugriff auf Dateien

Beispiel: Daten schreiben




```
Dim intFNr As Integer  ' Dateinummer
' Variablen für Beispiel
Dim strName As String
Dim datGebDat As Date
Dim bolGeschlecht As Boolean

' Freie Nummer für Dateizugriff besorgen
Let intFNr = FileSystem.FreeFile()

' Initialisierung mit Beispielwerten
Let strName = "Mike Müller"
Let datGebDat = #3/24/1998#
Let bolGeschlecht = False

' Datei öffnen (zum Schreiben)
Open "C:\Temp\doc3.txt" For Output Access Write As #intFNr

' Beispiel 1 schreiben (mit Variablen)
Write #intFNr, strName, datGebDat, bolGeschlecht
' Beispiel 2 schreiben (hier auch Typumwandlung sinnvoll)
Write #intFNr, "Ali Yilmaz", CDate("19.05.1987"), False

Close #intFNr  ' Datei schließen
```

Standarddialoge: Beispiel 10.04

Ziel

- Einfache Werte in Datei schreiben

Aufgabe

- Schreiben Sie den Text "Hallo Welt!" in eine Datei mit dem Pfad C:\Temp\HalloWelt.txt

Standarddialoge: Beispiel 10.05

Ziel

- Elemente eines Feldes in Datei schreiben

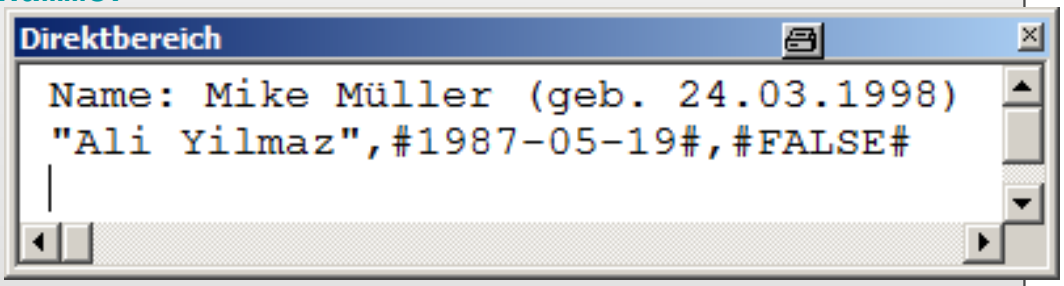
Aufgabe

- Definieren Sie einen Typ Person (mit Name, Vorname und Geburtsdatum)
- Deklarieren Sie ein Feld vom Typ Person, das mehrere Elemente hat
- Initialisieren Sie das Feld mit Angaben von drei Personen
- Speichern Sie alle Elemente des Feldes in einer Datei

Zugriff auf Dateien

Beispiel: Daten lesen

```
Dim intFNr As Integer  ' Dateinummer
' Variablen für Beispiel
Dim strName As String
Dim datGebDat As Date
Dim bolGeschlecht As Boolean
Dim strGanzeZeile As String

' Freie Nummer für Dateizugriff besorgen
Let intFNr = FileSystem.FreeFile()

' Datei öffnen (zum Lesen)
Open "C:\Temp\doc3.txt" For Input Access Read As #intFNr

' Beispiel 1 lesen mit Variablen
Input #intFNr, strName, datGebDat, bolGeschlecht
' Beispiel 2 lesen als Zeile
Line Input #intFNr, strGanzeZeile

Close #intFNr  ' Datei schließen

' Ausgabe
Debug.Print "Name: " & strName & " (geb. " & datGebDat & ")"
Debug.Print strGanzeZeile
```



Zugriff auf Dateien

Generelle Syntax

– Prüfen, ob Dateiende erreicht wurde

```
Let <bolVar> = FileSystem.EOF(<DateiNr>) ' Variante 1

Do Until FileSystem.EOF(<DateiNr>) ' Variante 2
 ' ...
Loop
```

```
' ...
Dim intFNr As Integer  ' Dateinummer
Let intFNr = FileSystem.FreeFile() ' Freie Nummer ermitteln

' Datei öffnen (zum Lesen)
Open "C:\Temp\Personen.txt" For Input Access Read As #intFNr
' Schleife bis Dateiende
Do Until FileSystem.EOF(intFNr)
  Input #intFNr, strName, datGebDat, bolWeibl
  Debug.Print strName &" (" datGebDat & ") " & bolWeibl
Loop
Close #intFNr  ' Datei schließen
```

Standarddialoge: Beispiel 10.06

Ziel

- Elemente eines Feldes aus einer Datei lesen

Aufgabe

- Definieren Sie einen Typ Person (mit Name, Vorname und Geburtsdatum)
- Deklarieren Sie ein dynamisches Feld vom Typ Person
- Lesen Sie alle Elemente des Feldes aus einer Datei
- Ausgabe des gesamten Feldes

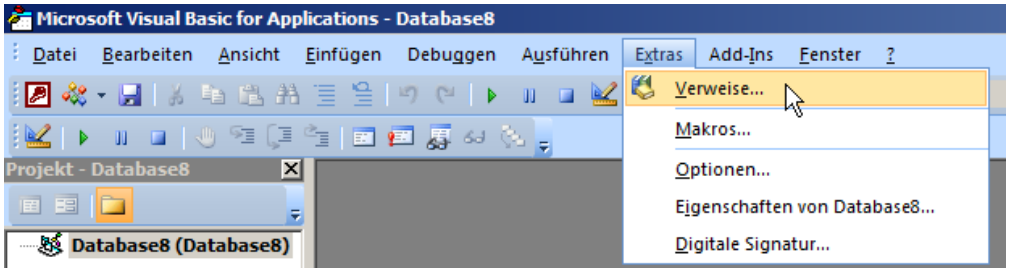
Inhalt

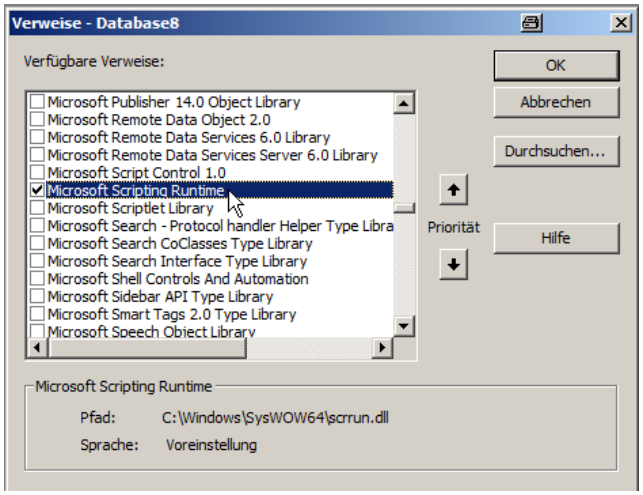
Inhalt

Zugriff mit Klassen aus MS Scripting Runtime

Erweiterte Möglichkeiten

- Bereitgestellt durch Bibliothek "Microsoft Scripting Runtime"
- Bibliothek muss eingebunden werden, um ihre Funktionen nutzen zu können
  - Im VBA-Editor > Extras > Verweise
  - Im Dialog "Verweise" den Eintrag "Microsoft Scripting Runtime" aktivieren > OK





Zugriff mit Klassen aus MS Scripting Runtime

FileSystem deklarieren und erzeugen

```
' Deklaration und Initialisierung
 Dim <FileSysObj> As FileSystemObject
 Set <FileSysObj> = New FileSystemObject
```

Öffnen einer vorhandenen Textdatei zum Lesen als Textstream (Einfachste Form)

- Generelle Syntax

```
' Deklaration und Initialisierung
 Dim <TxStreamObj> As TextStream
 Set <TxStreamObj> = <FileSysObj>.OpenTextFile(<Pfad>)
```

- Beispiel

```
' Beispiel: Vorhandene Textdatei für lesenden Zugriff öffnen
 Dim oFso As New FileSystemObject
 Dim oTxStream As TextStream

 Set oFso = New FileSystemObject
 Set oTxStream = oFso.OpenTextFile("c:\Temp\doc1.txt")
```

Zugriff mit Klassen aus MS Scripting Runtime

Öffnen einer Textdatei als TextStream

```
' Deklaration und Initialisierung
 Dim <FileSysObj> As FileSystemObject
 Set <FileSysObj> = New FileSystemObject
 Dim <TxStreamObj> As TextStream
 Set <TxStreamObj> = <FileSysObj>.OpenTextFile(<Pfad>, _
                          <Richtung>, <Erzeugen>, <Format>)
```

- Pfad: Angabe des Pfades zu einer Datei
- Richtung: Lesen (**`ForReading`**), zum Ändern (**`ForAppending`**) oder Schreiben (**`ForWriting`**)
- Erzeugen: Wahrheitswert, ob eine neue Datei angelegt werden soll, wenn sie noch nicht existiert bzw. überschrieben werden soll wenn Sie existiert
- Format: ASCII-Zeichensatz (**`TristateFalse`**), Unicode (**`TristateTrue`**) oder Systemstandard (**`TristateUseDefault`**)

Zugriff mit Klassen aus MS Scripting Runtime

Verwenden einer als TextStream geöffneten Datei

– Beispiel: Zeilenweises Lesen der Datei

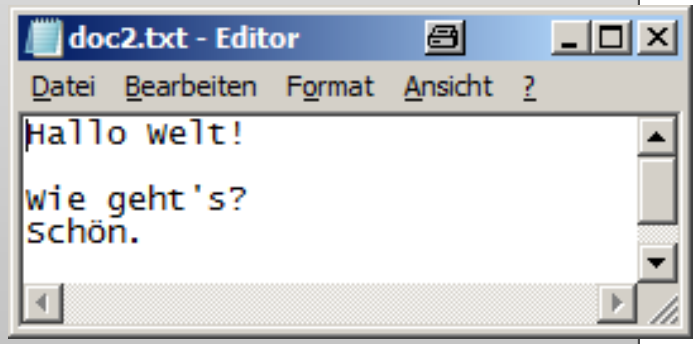
```
' Deklaration und Initialisierung
Dim oFso As New FileSystemObject
Dim oTxStream As TextStream

Set oFso = New FileSystemObject

' Öffnen einer Datei zum Lesen
Set oTxStream = oFso.OpenTextFile("c:\Temp\doc1.txt")

' Alles Lesen und im Direktbereich ausgeben,
' bis Ende des TextStreams erreicht ist
Do Until oTxStream.AtEndOfStream
 ' Ganze Zeile lesen und ausgeben im Direktbereich
 Debug.Print oTxStream.ReadLine
Loop
```

Zugriff mit Klassen aus MS Scripting Runtime

Verwenden einer als TextStream geöffneten Datei

– Beispiel: Schreiben einer neuen Datei

```
' Deklaration und Initialisierung
Dim oFso As New FileSystemObject
Dim oTxStream As TextStream

Set oFso = New FileSystemObject

' Öffnen einer Datei zum Schreiben,
' dabei ggf. neu anlegen bzw. überschreiben
Set oTxStream = oFso.OpenTextFile("c:\Temp\doc2.txt", _
                                  ForWriting, True, TristateTrue)

' Diverse Beispiele für Schreibzugriffe
Call oTxStream.Write("Hallo ")
Call oTxStream.WriteLine("Welt!")
Call oTxStream.WriteBlankLines(1)
Call oTxStream.WriteLine("Wie geht's?")
Call oTxStream.WriteLine("Schön.")
```



Inhalt

Inhalt

Einordnung

Rückblick

Ausgangspunkt

- Dateisystem
- Elemente im Dateisystem

Zugriff auf das Dateisystem

- Modul "FileSystem"
- FileSystem aus MS Scripting Runtime

Dialoge zur Datei- und Verzeichnisauswahl

- Standarddialoge
- FileDialog aus MS Office Object Library

Dateizugriff

- Grundlagen
- Zugriff auf Dateien
- Zugriff mit MS Scripting Runtime

Abschluss und Ausblick

Inhalt

Abschluss

Zugriff auf Dateisystem

- mit Modul "FileSystem" grundlegende Möglichkeiten
  - Elemente auflisten

```
' Generelle Syntax mit Angabe des gewünschten Inhalts
' z.B. vbDirectory, vbHidden, vbSystem
Let <strElement> = Dir(<Pfad>, <GewünschteInhalte>)
Let <strElement> = Dir() ' Nächstes (im vorherigen Pfad)
```

  - Weitere: Verzeichnisse anlegen, löschen, ...
- mit FileSystem-Klasse aus MS Scripting Runtime bestehen weitergehend Möglichkeiten z.B.
  - Zugriff auf Laufwerksinformation,
  - Kopieren von Verzeichnissen

Abschluss

Dialoge zur Auswahl von Dateien und Verzeichnissen

- sind sinnvoll, wenn vom Benutzer das Ziel zum Speichern oder Laden von Daten im Dateisystem selbst gewählt werden soll

Generelle Syntax

- Deklaration und Initialisierung

```
Dim <FileDlgObj> As Object
Set <FileDlgObj> = Application.FileDialog(<Zahl>)
```

- Konfiguration (z.B. Mehrfachauswahl)

```
<FileDlgObj>.AllowMultiSelect = True
```

- Anzeige

```
Let <intVar> = <FileDlgObj>.Show() ' Rückgabewert 0 = Abbruch
```

- Ergebnis in Collection "SelectedItems" enthalten

```
<FileDlgObj>.SelectedItems
```

Abschluss

Generelle Syntax (Fortsetzung)

– Ergebnis in Collection "SelectedItems" enthalten

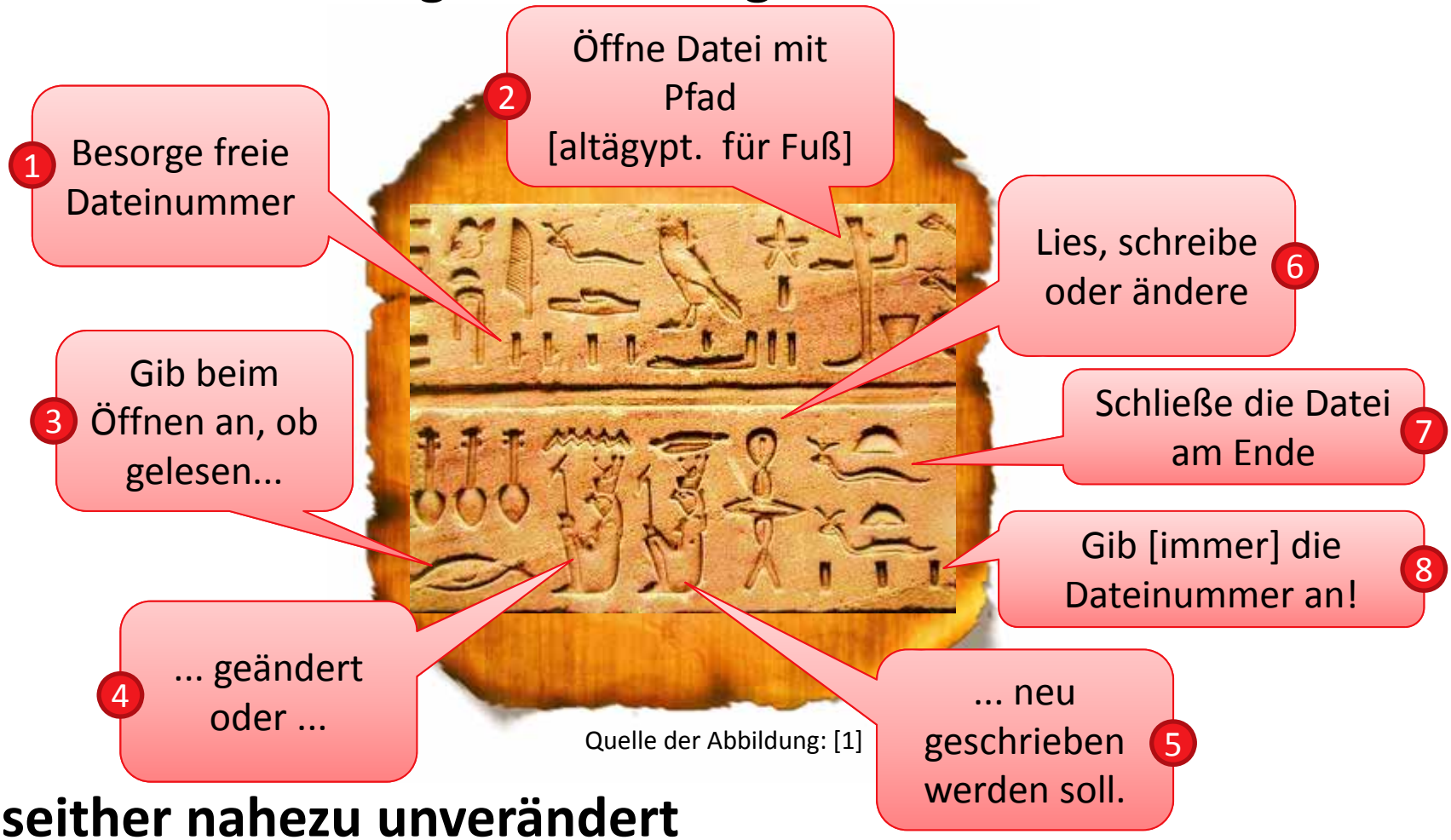
```
<FileDlgObj>.SelectedItems
```

Beispiel für Standarddialog zur Dateiauswahl

```
Dim intResult As Integer ' Rückgabewert
Dim i As Integer ' Schleifenvariable
Dim oFd As Object ' Variable für FileDialog
Set oFd = Application.FileDialog(3) ' Initial. als Dateiauswahl = 3

oFd.AllowMultiSelect = True ' Konfiguration, z.B. Mehrfachauswahl
Let intResult = oFd.Show ' Dialog anzeigen und Ergebnis merken

If intResult = 0 Then
 Exit Sub ' Abbruch durch Benutzer
End If

' Schleife über alle ausgewählten Dateien
For i = 1 To oFd.SelectedItems.Count
 Debug.Print oFd.SelectedItems(i)
Next
```

Grundlagen

Früheste Darstellung des Dateizugriffs in Basic



Quelle der Abbildung: [1]

... seither nahezu unverändert

Abschluss

Zugriff auf Dateien

- Freie Dateinummer ermitteln
- Öffnen einer Datei (verkürzte Form)
  - Pfad: Angabe des Pfades zur Datei
  - Modus: Lesen (Input), Schreiben (Output), Ändern (Append), ...
  - Zugriff: Lesen (Read), Schreiben (Write) oder Ändern (ReadWrite)
  - Dateinummer: Zuvor mit FreeFile() ermittelte Nummer
- Schließen einer Datei

Generelle Syntax

```
Let <intVar> = FileSystem.FreeFile()
```

```
Open <Pfad> For <Modus> Access
<Zugriff> As #<DateiNr>
```

```
Close #<DateiNr>
```

Beispiel

```
' Freie Nummer für Dateizugriff
Let intFNr = FileSystem.FreeFile()

' Datei öffnen (zum Schreiben)
Open "C:\Temp\doc3.txt" For Output _
  Access Write As #intFNr

' Datei verwenden
' ...

Close #intFNr  ' Datei schließen
```

Abschluss

Zugriff auf Dateien (Forts.)

- Datei schreiben (Write)

```
Write #<DateiNr>, <WertAusdr> ' Variante 1
Write #<DateiNr>, <WertAusdr1>, <WertAusdr2>, ... ' Variante 2
Write #<DateiNr>, ' Leere Zeile
```

- Datei lesen (Input)

```
Input #<DateiNr>, <Variable> ' Variante 1
Input #<DateiNr>, <Var1>, <Var2>, ... ' Variante 2
```

- Datei zeilenweise lesen (Line Input)

```
Line Input #<DateiNr>, <StringVariable>
```

Beispiel (Schreibzugriff)

```
'...
' Beispiel 1 schreiben (mit Variablen)
Write #intFNr, strName, datGebDat, bolGeschlecht
' Beispiel 2 schreiben (hier auch Typumwandlung sinnvoll)
Write #intFNr, "Ali Yilmaz", CDate("19.05.1987"), False
'...
```

Ausblick

06 – Zusammenfassung

05 – Debugger und Testen

04 – Fortgeschrittene Konzepte

04.A	04.B	04.C
Oberflächen (Teil 1) Elemente und Eigenschaften	Oberflächen (Teil 2) Ereignisverarbeitung	Zugriff auf Dateisystem und Anwendungen

03 – Grundkonzepte

03.A	03.B	03.C	03.D	03.E
Wert Ausdruck Variable Konstante Datentyp	Bedingte Ausführung/ Verzwei-gungen	Schleifen	Felder Mengen	Prozedur Funktion Modul

02 – Grundlagen der Programmierung

01 – Grundbegriffe der Wirtschaftsinformatik

Quellen

[1] A☮ineko: Hiéroglyphes, temple de Komombo. (März 2000), Lizenz: Creative Commons-Lizenz Namensnennung-Weitergabe unter gleichen Bedingungen 1.0 US-amerikanisch (nicht portiert), http://commons.wikimedia.org/wiki/File:Egypt_Hieroglyphe2.jpg?uselang=de

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Wirtschaftsinformatik 1
LE 10 – Zugriff auf das Dateisystem

Prof. Dr. Thomas Off

http://www.ThomasOff.de/lehre/beuth/wi1