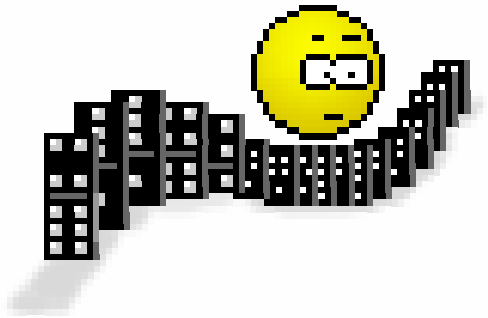
Dirk Alicke
Südwestrundfunk Stuttgart
www.swr.de

IKS
INFORMATIONS- UND KOMMUNIKATIONSSYSTEME

Datenaustausch zwischen Notes und Excel, Word und PowerPoint unter Nutzung der OLE-Automation

-

eine komplette Anleitung für Einsteiger



SWR

IKS

INFORMATIONS- UND KOMMUNIKATIONSSYSTEME

Was erwartet Sie?

- Voraussetzungen für den Datenaustausch
- Wirtschaftliche Gesichtspunkte
- Übersicht über Schnittstellen
- VBA-Code nach LotusScript übernehmen
- Excel, Word, PowerPoint
  Export, Import
- RichText-Felder (Ansätze zur Behandlung)
- Konstanten in MS-Applikationen
- Lotus Symphony

Software und Kenntnisse

Software

- Lotus Notes ab Version 4.x
- MS-Word
- MS-Excel
- MS-PowerPoint

Kenntnisse

- LotusScript
- Vorkenntnisse in VBA (vorteilhaft aber nicht notwendig)

Wirtschaftliche Gesichtspunkte

Beispiel

Die Sekretärin Schreibeschnell erstellt jeden Monat in Excel ein Diagramm. Dafür muss sie händisch aus 10 Anwendungen Daten übertragen und aufbereiten. Pro Anwendung benötigt sie 2 Stunden und die Personalkosten betragen pro Stunde 60,- €. Durch ein Script könnte diese Aufgabe auf einen Zeitaufwand von insgesamt 10 Minuten reduziert werden. Der Entwicklungsaufwand beträgt dabei 9900 €.

Wirtschaftliche Gesichtspunkte

Aufwand vor der Anpassung

$$A_{alt} = 10\frac{Anw}{Mon} * 2\frac{h}{Anw} * 60\frac{€}{h}$$

$$\underline{A_{alt} = 1.000\frac{€}{Mon}}$$

Aufwand nach der Anpassung

$$A_{neu} = \frac{1}{6}\frac{h}{Mon} * 60\frac{€}{h}$$

$$\underline{A_{neu} = 10\frac{€}{Mon}}$$

Einsparung pro Monat

$$E_{Mon} = 1.000\frac{€}{Mon} - 10\frac{€}{Mon}$$

$$\underline{E_{Mon} = 990\frac{€}{Mon}}$$

Return of Invest

$$R = \frac{9.900€}{990\frac{€}{Mon}}$$

$$\underline{\underline{R = 10Mon}}$$

IKS

INFORMATIONS- UND KOMMUNIKATIONSSYSTEME

Schnittstellen (API)

- **API … „Application Programming Interface“ (Schnittstelle zur Anwendungsprogrammierung)**
  - Schnittstelle, die ein Betriebssystem oder ein Softwaresystem zur Verfügung stellt
  - DLL … Dynamic Link Libraries

Ausführliche Infos:
http://de.wikipedia.org/wiki/Programmierschnittstelle

Schnittstellen (DDE)

- **DDE … „Dynamic Data Exchange“**
  - Kanal zu einem DDE-fähigen Programm öffnen und Befehle übergeben
  - langsam und kompliziert
  - Programme verhalten sich nicht immer „richtig“ (Twainschnittstellenproblem)
  - beide Programme laufen sichtbar nebeneinander her

Ausführliche Infos:
http://de.wikipedia.org/wiki/Dynamic_Data_Exchange

Schnittstellen (COM)

- **COM … „Component Object Model“**
  - von Microsoft entwickelt
  - Client / Server-Prinzip
  - COM-Client instanziert eine COM-Komponente in einem COM-Server (DLL oder ausführbares Programm) und nutzt die Funktionalität des Objektes
  - COM ist die Basis für OLE-Automation und ActiveX
  - ist sprach-, versions-, plattform-, ortsunabhängig, automatisierend und objektorientiert

Ausführliche Infos:
http://de.wikipedia.org/wiki/Component_Object_Model

Schnittstellen (OLE)

- **OLE … „Object linking and / or embedding“**
  - von Microsoft entwickelt
  - sichtbar und unsichtbar
  - Zugriffslogik gegenüber DDE geändert
  - verlinken … Verweis auf das Objekt
  - einbetten … Kopie des Originals
  - Server-Anwendung
  - Objekte, Methoden und Eigenschaften stehen zur Verfügung

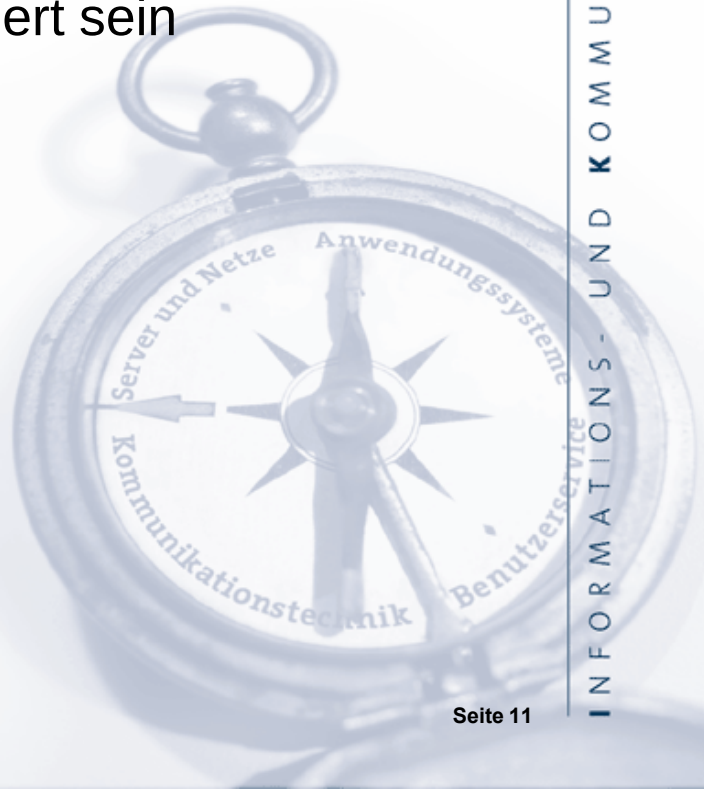
Ausführliche Infos:
http://de.wikipedia.org/wiki/Object_Linking_and_Embedding

OLE – Versionen

- **OLE 1.0 und OLE 2.0**
  - OLE 1.0 startet das gesamte Programm
  - OLE 2.0 startet nur den Kernel (ca. 60 – 80% der Software)
  - keine „echte“ Fernsteuerung
- **OLE - Automation**
  - echte Fernsteuerung
- **OA – Office Automation**
  - anwendungsübergreifende Programmierung innerhalb von MS Office
  - neuer Begriff für OLE-Automation

Nachteile von OLE

- **Nachteile von OLE**
  - ist abhängig von den verwendeten Programmversionen (Softwareupdates und Sprachänderungen können Probleme bereiten)
  - Server-Anwendung muss installiert sein



Registry

- **HKEY_CLASSES_ROOT**
  - Programme, die per OLE angesprochen werden können
    z.B. Excel: Excel.Application
    CLSID ist ein weltweit eindeutiger Schlüssel
    {00024500-0000-0000-C000-000000000046}

- **HKEY_LOCAL_MACHINE\SOFTWARE\Classes\CLSID**
  - LocalServer32 … Excel.EXE mit Parameter /Automation
  - Nicht das komplette Excel wird gestartet, sondern nur der Funktionsteil (ca. 80%).

CreateObject

CreateObject

- Syntax: Set Object = CreateObject(ClassName)
- neue Instanz einer Anwendung generieren
- Anwendung als Objekt zur Verfügung stellen

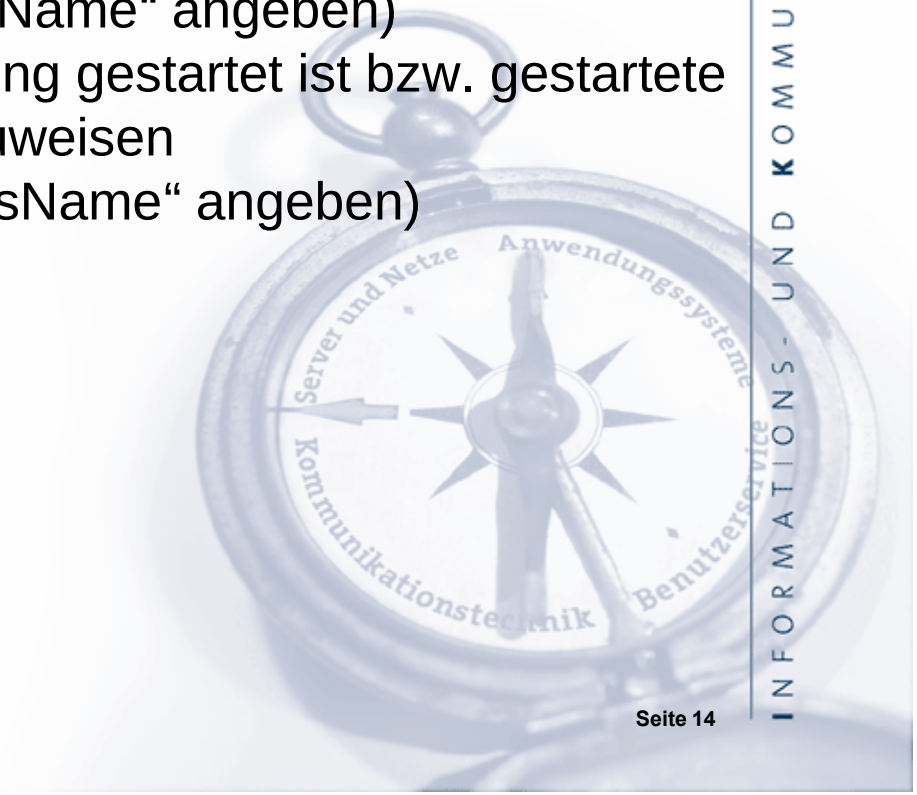
Beispiele

~~Set wdApp = CreateObject("Word.Application.10")~~
Set wdApp = CreateObject("Word.Application")
Set xlApp = CreateObject("Excel.Application")
Set frApp = CreateObject("FineReader.Application")
Set ppApp = CreateObject("PowerPoint.Application")
Set ieApp = CreateObject("InternetExplorer.Application")
Set olApp = CreateObject("Outlook.Application")
Set acApp = CreateObject("Access.Application")

GetObject

GetObject

- Syntax: Set Object = GetObject(PathName, ClassName)
- Anwendung steht als Objekt zur Verfügung
- Anwendung anhand einer Datei starten
  (nur Parameter „PathName“ angeben)
- Prüfen, ob eine Anwendung gestartet ist bzw. gestartete Anwendung als Objekt zuweisen
  (nur Parameter „ClassName“ angeben)



Die erste OLE-Verbindung

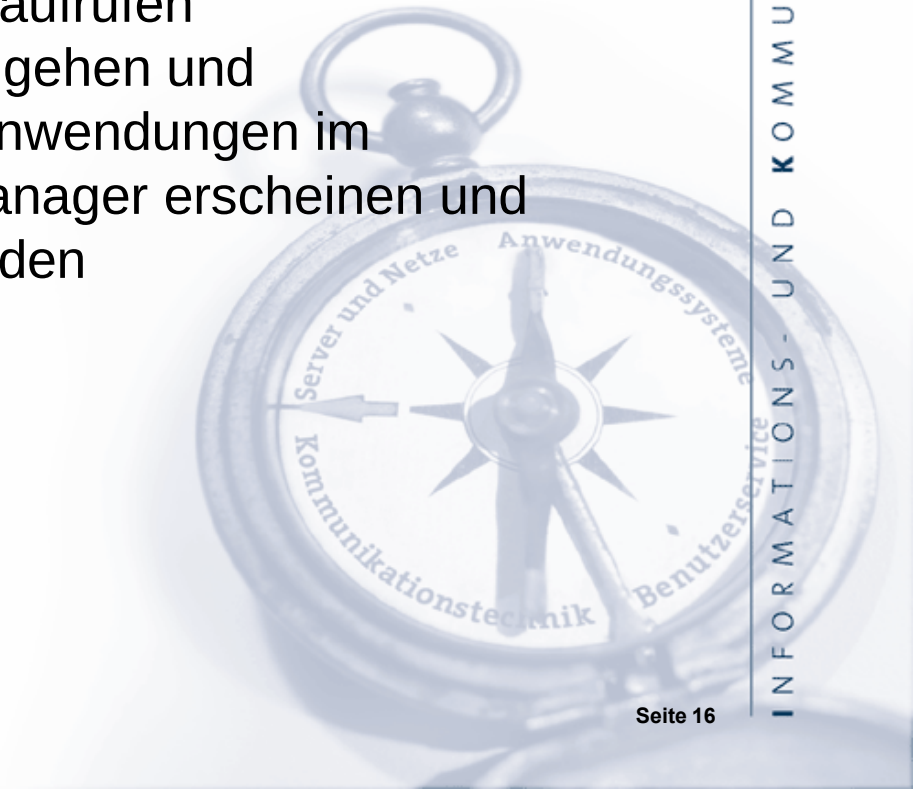
```
‘ OLE-Verbindung zu Excel erzeugen
Set xlApp = CreateObject("Excel.application")
‘ existiert das Objekt?
If xlApp is Nothing Then Exit Sub
xlApp.Visible = True                ‘ Excel anzeigen
‘ xlApp.Visible = False             ‘ Excel nicht anzeigen
xlApp.Workbooks.add                 ‘ neue Excel Mappe erstellen
```

```
‘ W I C H T I G, falls Visible auf False gesetzt wurde, muss die Applikation zum Schluss beendet werden !
xlApp.Quit      ‘ Excel beenden
```

DEMO: erste OLE-Verbindung

Agent „DEMO 1\01. Die erste OLE-Verbindung“

1. Debugger einschalten!
2. Windows-Taskmanager aufrufen
3. Agent schrittweise durchgehen und
4. beobachten, wann die Anwendungen im Frontend und im Taskmanager erscheinen und wann diese beendet werden



DEMO: GetObject / CreateObject

Agent „DEMO 1\02. Excel - GetObject / CreateObject“

1. mit GetObject prüfen, ob Excel gestartet wurde
2. falls Excel noch nicht gestartet wurde, Excel mit CreateObject starten

Anmerkungen zu „Visible = True“

Die Anwendung wird sofort angezeigt (Visible = True)

Vorteile:

- User sieht, dass sich etwas tut (Zappeleffekt)
- Bei einem Fehler im Script kann die Anwendung vom User geschlossen werden (es bleiben keine Prozesse offen).

Nachteile:

- User kann die Kommunikation stören (fehleranfälliger).
- Kommunikation zwischen den Anwendungen ist langsamer.

Anmerkungen zu „Visible = False“

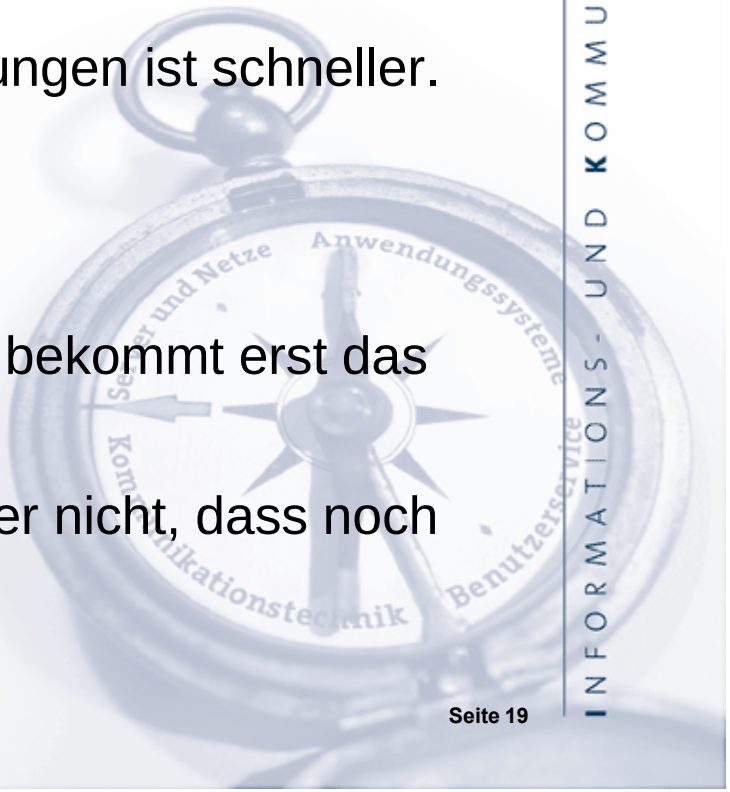
Die Anwendung wird nicht angezeigt (Visible = False):

Vorteile:

- User kann die Kommunikation nicht stören.
- Kommunikation zwischen den Anwendungen ist schneller.

Nachteile:

- User sieht nicht, dass sich etwas tut, er bekommt erst das Ergebnis zu Gesicht.
- Bei einem Fehler im Script sieht der User nicht, dass noch versteckte Prozesse offen sind.



Vorhandene Office-Makros verwenden

Word starten und Makro ausführen

Sinnvoll, wenn das Makro schon vorhanden ist.
Syntax: wdApp.Application.Run "Makro01 "

Vorteile

- User muss nicht mehrere Aktionen ausführen
- wenig Code im Notes
- schnelle Umsetzung, wenn das Makro vorhanden ist

Nachteile

- User braucht die Vorlage mit dem Makro
- Fehler müssen im Makro behandelt werden

DEMO: Word - Makro

Agent „DEMO 1\03. Word-Makro starten“

1. Word starten
2. Makro „Makro01“ ausführen

```
Sub Makro01()
    With Selection.Font
        .Name = "Arial"
        .Size = 20
        .Color = wdColorRed
    End With
    Selection.TypeText Text:="Dieser Text wurde vom Word-Makro ""Makro01"" geschrieben."
End Sub
```

Makrorecorder Teil 1

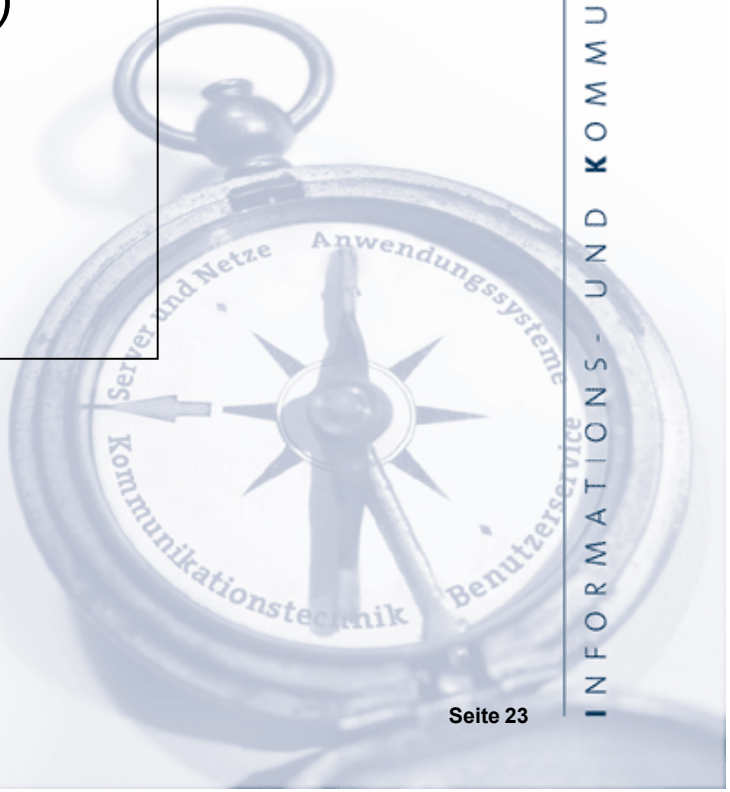
1. Makro mit Recorder aufzeichnen

```
Sub Makro1()
    Selection.Borders(xlDiagonalDown).LineStyle = xlNone
    Selection.Borders(xlDiagonalUp).LineStyle = xlNone
    Selection.Borders(xlEdgeLeft).LineStyle = xlNone
    Selection.Borders(xlEdgeTop).LineStyle = xlNone
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
    Selection.Borders(xlEdgeRight) = xlNone
End Sub
```

Makrorecorder Teil 2

2. Ballast abwerfen

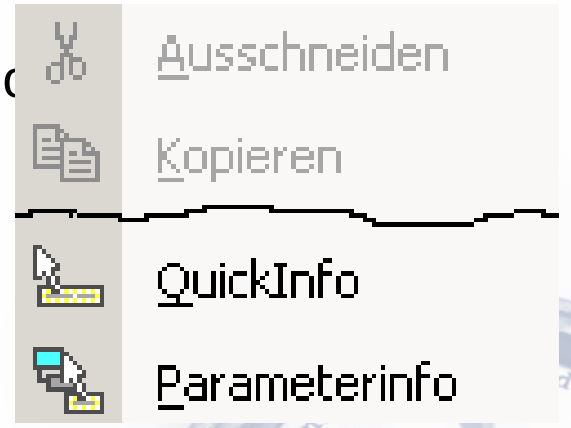
```
Sub Makro1()
    With Selection.Borders(xlEdgeBottom)
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
End Sub
```



Makrorecorder Teil 3

3. Konstanten mit Quickinfo anzeigen lassen (rechte Maustaste auf Konstantennamen)

```
Sub Makro1()
    With Selection.Borders(xlEdgeBo
        .LineStyle = xlContinuous
        .Weight = xlThin
        .ColorIndex = xlAutomatic
    End With
End Sub
```




```
With Selection.Borders(xlEdgeBottom)
    .LineStyle = xlConti
    .Weight = xlThin
    .ColorIndex = xlAutomatic
End With
```

xlEdgeBottom = 9

Makrorecorder Teil 4

4. VBA-Konstanten ersetzen

```
Sub Makro1()
        With Selection.Borders(9)
                .LineStyle = 1
                .Weight = 2
                .ColorIndex = -4105
        End With
End Sub
```

5. Code nach Notes kopieren und syntaktisch anpassen

```
xlApp.Range("A1:J20").Select
With xlApp.Selection.Borders(9)
    .LineStyle = 1
    .Weight = 2
    .ColorIndex = -4105
End With
```

DEMO: Excel - Makrorecorder

Agent „DEMO 1\04. Excel – Makrorecorder“

1. Makro im Excel aufzeichnen
2. Konstanten mit Hilfe von QuickInfo ersetzen
3. Code in den LotusScript-Agenten kopieren
4. Syntax anpassen

Konstanten

(Access 97, Excel 97, 5 und 7, Office 97, Office Binder 97, Outlook 97, VBA , Word 97)

Konstanten stehen in der VBA-Hilfe unter Konstanten

(Options)

```
%INCLUDE "ConstAccess97.lss"          ' 597 Const
%INCLUDE "ConstExcel97.lss"           ' 1266 Const
%INCLUDE "ConstOffice97.lss"          ' 794 Const
%INCLUDE "ConstOfficeBinder97.lss"    ' 11 Const
%INCLUDE "ConstOutlook2000.lss"       ' 251 Const
%INCLUDE "ConstPowerPoint2002.lss"    ' 816 Const
%INCLUDE "ConstVB.lss"                ' 279 Const
%INCLUDE "ConstVBA.lss"               ' 246 Const
%INCLUDE "ConstWord2000.lss"          ' 2395 Const
```

Excel-Start in eine Funktion auslagern

```
Function Excel_Start(xlApp As Variant, Anzeigen As Variant) As Variant
‘ Anzeigen ... True oder False … soll Excel angezeigt werden?
    Excel_Start = False
    Set xlApp = Nothing
    Set xlApp = CreateObject("Excel.application")

    ‘ existiert die OLE-Verbindung
    If xlApp is Nothing Then Exit Function
    xlApp.Visible = anzeigen
    xlApp.Workbooks.add
    Excel_Start = True
End Function
```

Excelzellen lesen und schreiben

Schreiben

```
xlApp.Cells( ZeilenNummer, SpaltenNummer).Value = Wert
```

Beispiel:

```
For iZeile = 1 To 10
    For iSpalte = 1 To 20
      xlApp.Cells( iZeile,iSpalte).Value = iZeile * iSpalte
   Next
Next
```

Lesen

```
Value = xlapp.Cells(ZeilenNummer, SpaltenNummer).Value
```

DEMO: Excel – Ansicht

Agent „DEMO 1\05. Excel - Ansicht nach“
Ansicht „DEMO 1\05. Excel - Ansicht nach“

1. Alle Einträge der aktuellen Ansicht werden nach Excel geschrieben.

DEMO: Excel - Import

Agent „DEMO 1\06. Excel - Import F1“
Ansicht „DEMO 1\06. Excel - Import F1“

1. Excel Datei „F1.xls“ öffnen
2. richtigen Tab wählen
3. neues Dokument erstellen
4. auslesen der einzelnen Werte und eintragen ins Notes-Dokument

Spaltenzahl in –buchstabe konvertieren

```
Function xlSZ2SB(Spaltenzahl As Long) As String
        If Fix((Spaltenzahl - 1) / 26) = 0 Then
                SP1 = ""
        Else
                SP1 = Chr(Fix((Spaltenzahl - 1) / 26) + 64)
        End If
        If Spaltenzahl Mod 26 = 0 Then
                SP2 = "Z"
        Else
                SP2 = Chr(Spaltenzahl Mod 26 + 64)
        End If
        xlSZ2SB = SP1 & SP2
End Function
```

Test: SpaltenZahlen2SpaltenBuchstaben

Excel - Tipp

Automatische Anpassung der Spaltenbreite einstellen:
xlApp.Range(xlRange).EntireRow.AutoFit

Automatische Anpassung der Zeilenhöhe einstellen:
xlApp.Range(xlRange).EntireColumn.AutoFit

Excel - Performance

Änderungen an mehreren Zellen im Excel sollten immer über Range erfolgen und nicht über die Zelleneigenschaften.

```
‘ Formatierung über Range
xlRange = "A1:C1“ ‘ Range A1:C1
xlApp.Range(xlRange).Font.Bold = True

‘ Formatierung über Cells
xlApp.Cells(1, 1).Font.Bold = True
xlApp.Cells(1, 2).Font.Bold = True
xlApp.Cells(1, 2).Font.Bold = True
```

DEMO: Excel - Range

Agent „DEMO 1\07. Excel - Range“

1. die Zellen A1 bis A500 werden mit Cells formatiert
2. die Zellen B1 bis B500 werden mit Range formatiert
3. mit GetThreadInfo wird die Zeit gemessen

Excel - Chart

Wie würden Sie vorgehen, wenn Sie ein Excel-Diagramm mit Lotusscript erstellen müssten?

```
xlRange="$A$1:$C$10"
xlApp.Charts.Add
xlApp.ActiveChart.ChartType = 51
xlApp.ActiveChart.SetSourceData xlApp.Sheets("Sheet1").Range("A1:C10"), 2
xlApp.ActiveChart.SeriesCollection(1).Name = "="“Mein Chart""""
xlApp.ActiveChart.Location 2, "Sheet1“
xlApp.ActiveChart.HasTitle = True
xlApp.ActiveChart.ChartTitle.Characters.Text = "Notes Demo“
xlApp.ActiveChart.Axes(1,1).HasTitle = True
xlApp.ActiveChart.Axes(1,1).AxisTitle.Characters.Text = "X“
xlApp.ActiveChart.Axes(2,1).HasTitle = True
xlApp.ActiveChart.Axes(2,1).AxisTitle.Characters.Text = "Y“
xlApp.ActiveChart.ApplyDataLabels 2, True
xlApp.ActiveChart.HasDataTable = False
xlApp.ActiveSheet.Shapes("Chart 1").ScaleHeight 1.46, 0,0
```

DEMO: Excel - Diagramm

Agent „DEMO 1\08. Excel - Diagramm“

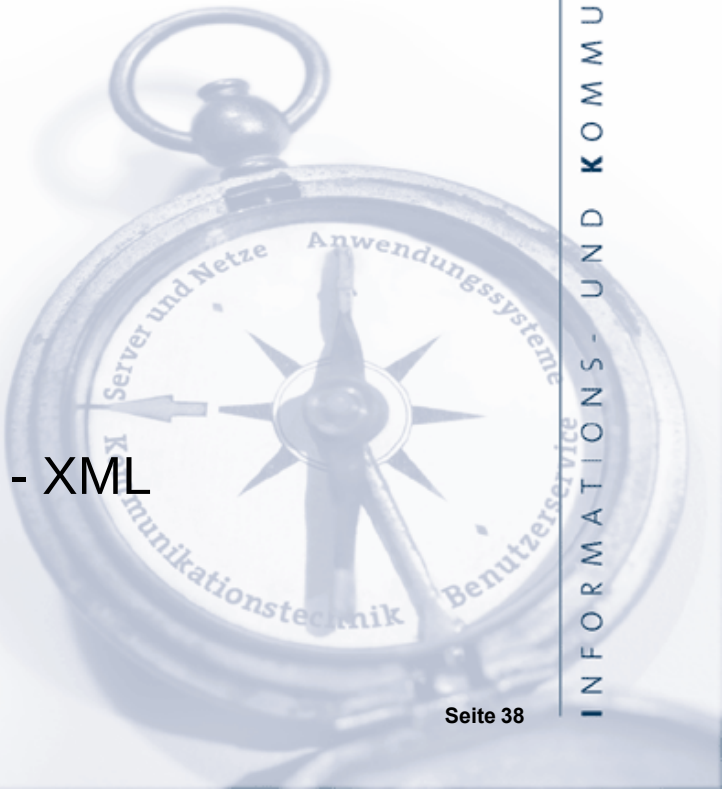
1. Datenreihe mit Zufallszahlen erstellen (Einnahmen und Ausgaben)
2. aus dieser Datenreihe ein Linien-diagramm erzeugen

Excel – Performance

Datenübergabe mit Hilfe einer XML-Datei:

- XML-Datei schreiben und im Excel öffnen (ab Excel 2002)
- Diagramme können so nicht erzeugt werden
- Diagramme anschließend per OLE erzeugen
- wesentlich schneller

DEMO: DEMO 1\09. Excel - XML



Ansätze für Richtext-Felder

Frontend-Dokument

- im Editmode per Zwischenablage
  Demo: DEMO 2\10. Word - Export (RTF)

Backend-Dokument

- eventuell als XML exportieren und transformieren
- eventuell Richtextklassen ab Notes 6.x benutzen

DEMO: Word - Querformat

Agent „DEMO 2\11. Word - Export (Rezept)“
Ansicht „DEMO 2\11. Word - Export “

1. alle Dokumente in der Ansicht Rezepte exportieren
2. Worddokument im Querformat und 2-spaltig einstellen

Microsoft PowerPoint

```
Set ppApp = CreateObject(″PowerPoint.Application")
ppApp.Presentation.Add 1, 1

With ppApp.ActiveWindow.Selection
    .SlideRange.Shapes("Rectangle 2").Select
    .ShapeRange.TextFrame.TextRange.Select
    .ShapeRange.TextFrame.TextRange.Characters
(1,0).Select
    .TextRange.Text = Sessiontitle
End With
```

DEMO: PowerPoint – Export

Agent „DEMO 2\12. PowerPoint – Export“
Ansicht „DEMO 2\12. PowerPoint – Export“

1. alle Dokumente in der Ansicht „DEMO\PowerPoint Export“ exportieren
2. jedes Dokument ist dabei eine neue PowerPoint-Seite

DEMO: PowerPoint – Import

Agent „DEMO 2\13. PowerPoint – Import“
Ansicht „DEMO 2\13. PowerPoint – Import“

1. diese Powerpoint-Präsentation wird importiert
2. Import funktioniert über die Zwischenablage
3. importiert wird in ein RichText-Feld

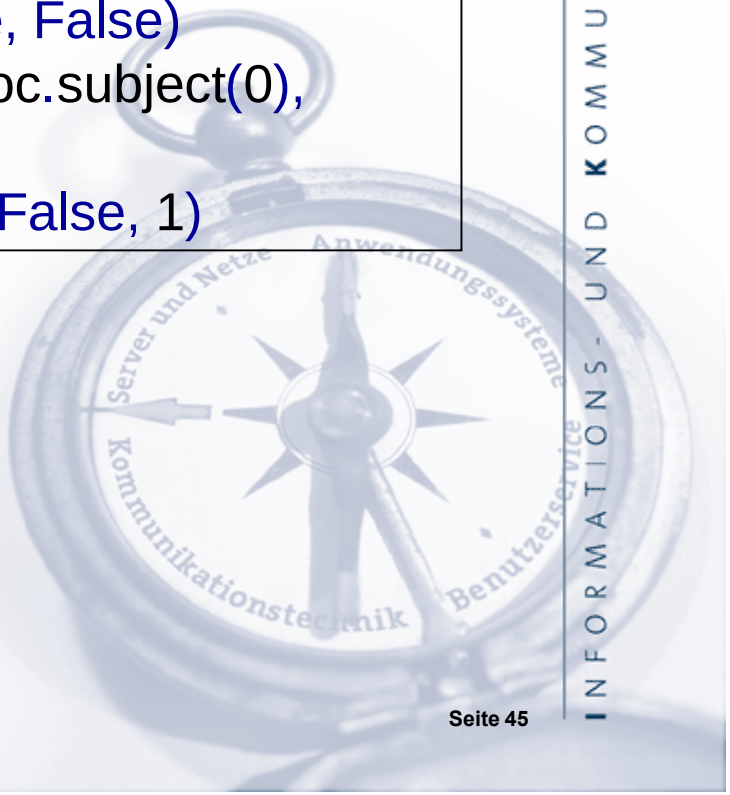
Vergleich Word / Excel / PowerPoint

Word	Excel	PowerPoint
Document	Workbook	Presentation
Page	Worksheet	Slide
Paragraph	Range	Text Block
Text Position / Cursor	Cell	Text Position / Cursor

Microsoft Project

```
Set projectApp = CreateObject("msproject.Application")

Call projectApp.filenew(False, "", False, False)
Call projectApp.settaskfield("Name", doc.subject(0),
False, True, 1)
Call projectApp.selectrow(projidcount, False, 1)
```



Lotus Symphony

- ab Notes 8.0.1

1. ServiceManager ist das Backend Objekt in Lotus Symphony
2. Desktop ist das Frontend Objekt in Lotus Symphony
3. danach wird erst die Anwendung instanziert

```
Set SM = CreateObject(“com.sun.star.ServiceManager”)
Set Desk = SM.createInstance(“com.sun.star.frame.Desktop")
```

DEMO: Lotus Symphony Writer

Agent „DEMO 2\14. Lotus Symphony Writer“

```
Dim args()
Set wApp = Desk.loadComponentFromURL
      ("private:factory/swriter", "_blank", 0, args)
Set wText = wApp.getText()  ‘ Handel auf Textbereich
Set Cur = wText.createTextCursor()  ‘ Cursor Position
Call wText.insertString(Cur, “EntwickerlCamp 2009", False)
```

DEMO: Lotus Symphony Spreadsheets

Agent “DEMO 2\15. Lotus Symphony Calc”

```
Set CalcApp =
      Desk.loadComponentFromURL("private:factory/scalc",
      "_blank", 0, args)
Set wsheet = CalcApp.Sheets.getByName("A")
Set cell = wsheet.getCellByPosition(3,6)
Call cell.setString(“EntwicklerCamp 2009")

‘ Hinweis: Zellenindex beginnt bei 0
```

DEMO: Lotus Symphony Presentations

Agent “DEMO 2\16. Lotus Symphony Presentations”

```
Set presApp = Desk.loadComponentFromURL_
("private:factory/simpress", "_blank", 0, args)

Set pres = presApp.getDrawPages()

Set Slide = pres.getByIndex(0)
Slide.layout = 1

Set title = Slide.getbyindex(0)
Set TitleText = title.getText()
Set Cur = TitleText.createTextCursor()
Call TitleText.insertString(Cur, “EntwicklerCamp 2009", False)
```

Quellen

Hilfen

VBA
Notes

WEB-Links

msdn.microsoft.com/vbasic/
www.martinscott.com/dominosupersearch2.nsf/Search?OpenForm

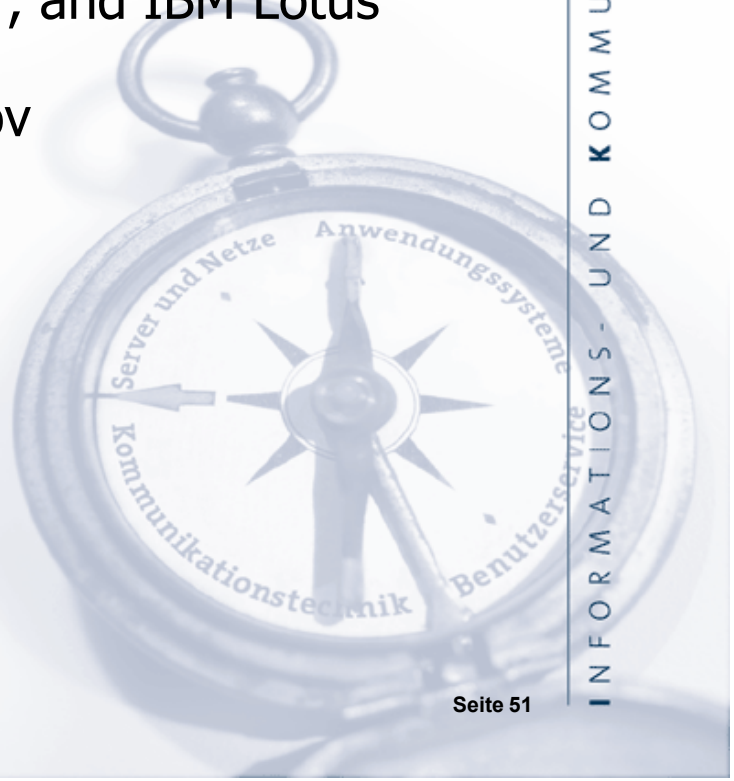
Foren

www.dominoforum.de
www.atnotes.de

Quellen

Vortrag:

Lotusphere 2009
"JMP205: Integration of IBM Lotus Notes and Lotus Domino with Microsoft Office, .NET, and IBM Lotus Symphony"
von John D. Head und Alex Kassbov



SWR

IKS

INFORMATIONS- UND KOMMUNIKATIONSSYSTEME

Vielen Dank für Ihre Aufmerksamkeit.

Jetzt sind Sie an der Reihe.