assono

EntwicklerCamp 2016
Track 2, Session 8:
LotusScript **extrem!**

Gelsenkirchen, 13. April 2016

Innovative Software-Lösungen

www.assono.de

assono

Thomas Bahn

- Diplom-Mathematiker, Universität Hannover
- seit 1997 entwickle ich mit Java und relationalen Datenbanken
- seit 1999 mit Notes/Domino zu tun: Entwicklung, Administration, Beratung und Schulungen
- regelmäßiger Sprecher auf nationalen und internationalen Fachkonferenzen zu IBM Lotus Notes/Domino und Autor für THE VIEW

tbahn@assono.de
www.assono.de/blog
04307/900-401

assono
IT-Consulting & Solutions

01 – High Resolution Timer – Variablendeklaration

- Frühe und späte Bindung von Variablen
- Gibt es einen messbaren Unterschied?

```
Dim i As Long
i = 0
For i = 1 To 10(
    i = i + 1
Next
```

```
Dim j As Variant
j = 0
For j = 1 To 1000000
    j = j + 1
Next
```

```
k = 0
For k = 1 To 1000000
    k = k + 1
Next
```

- Variant/keine Deklaration braucht über 20mal so lang

assono

01 – High Resolution Timer – Variablendeklaration

- Man spricht auch von „early binding“ (früher Bindung) und „late binding“ (später Bindung)
- Bei früher Bindung findet der Compiler die Fehler, bei später Bindung der Benutzer!
- Verschreiber bei den Variablennamen…
- Deshalb immer **`Option Declare`**!

- Welchen Typ haben diese Variablen?

```
Dim i, j, k As Integer
```

assono

01 – High Resolution Timer – Variablendeklaration

- Man spricht auch von „early binding“ (früher Bindung) und „late binding“ (später Bindung)
- Bei früher Bindung findet der Compiler die Fehler, bei später Bindung der Benutzer!
- Verschreiber bei den Variablennamen…
- Deshalb immer **`Option Declare`**!

- Welchen Typ haben diese Variablen?

```
Dim i, j, k As Integer
```

- nur k ist Integer, i und j sind **Variants**!

assono

01 – High Resolution Timer – genaue Messungen

- Und wie misst man das?
- mit **GetThreadInfo(LSI_THREAD_TICKS)**

```
startTicks = GetThreadInfo(LSI_THREAD_TICKS)
[…]
endTicks = GetThreadInfo(LSI_THREAD_TICKS)
time = (endTicks - startTicks)/_
    GetThreadInfo(LSI_THREAD_TICKS_PER_SEC)
```

assono

02 – Leere NotesDocumentCollection erzeugen

- früher:

```
Set coll = db.Search(|Form = "gibbetNit"|, Nothing, 1)
```

- ab Notes 8

```
Set coll = db.CreateDocumentCollection
```

- undokumentierte Funktion, aber es gibt eine Technote

assono

03 – StampAllMulti

- viele Dokumente gleichzeitig ändern:

```
Call coll.StampAll("Änderer", session.UserName)
```

- viele Werte in vielen Dokumenten gleichzeitig ändern (ab Notes 8.5):

```
Call coll.StampAllMulti(docWithValues)
```

- schreibt Änderungen sofort in die Dokumente
- gibt es beide für NotesDocumentCollection und NotesViewEntryCollection

assono

04 – Fehlerbehandlung – Variante A

- minimale „Fehlerbehandlung“: Ausgabe für Benutzer

```
Dim a As Integer

On Error GoTo errorHandler

' do something (wrong)
a = 0
Print 1/a
Exit Sub

errorHandler:

MessageBox "Fehler #" & Err & " in Zeile " & Erl & _
    " aufgetreten: " & Chr$(10) & Error$, MB_ICONSTOP

End ' some jump to Exit Sub, but status may be inconsistent!
```

assono

04 – Fehlerbehandlung – Variante B

- etwas mehr

```
errorHandler:

errorMessage = "Fehler #" & Err & " in Zeile " & Erl &_
    " aufgetreten: " & Chr$(10) & Error$ & Chr$(10) & Chr$(10)
errorMessage = errorMessage & "Routine " &_
    GetThreadInfo(LSI_THREAD_PROC) & Chr$(10)
errorMessage = errorMessage & "Aufrufer " &_
    GetThreadInfo(LSI_THREAD_CALLPROC) & Chr$(10)

If Not session.IsOnServer() Then ' LSI_Info is not thread-safe
    stack = LSI_Info(14)
    If IsArray(stack) Then stack = Join(stack, Chr$(10))
    errorMessage = errorMessage & "Stack " & stack
End If

MessageBox errorMessage, MB_ICONSTOP, "Fehler aufgetreten"

End
```

assono

04 – Fehlerbehandlung – GetThreadInfo und LSI_Info

- GetThreadInfo liefert viele Kontextinformationen
- LSI_Info ist „böse“ (zumindest auf dem Server)
  - nicht thread-safe
  - kann zu Abstürzen führen
- **Aber**: leider keine Alternative für `LSI_Info(14)` (Call-Stack) verfügbar

- notes.ini: DEBUG_LS_DUMP=1 (Technote)
  - Call-Stack im NSD-Dump im Moment des Absturzes

assono

04 – Fehlerbehandlung – Variante C

- diesmal zwei Operationen

```
If Not IsDebugMode Then On Error GoTo errorHandler

' do something (wrong)
a = 0
Print 1/a

Kill "B:\Test.txt"
Messagebox "Datei erfolgreich gelöscht", MB_ICONINFORMATION
Exit Sub
```

assono

04 – Fehlerbehandlung – Variante C

- jetzt wirklich „behandeln“:

```
errorHandler:

Select Case HandleError()
    Case ERROR_HANDLER_RESUME_NEXT
        Resume Next

    Case ERROR_HANDLER_RETRY
        Resume 0

    Case Else
        End
End Select
```

assono

04 – Fehlerbehandlung – Variante C

- HandleError(), 1. Teil

```
HandleError = ERROR_HANDLER_END

Select Case Err
    Case 11 ' Division by zero
        HandleError = ERROR_HANDLER_RESUME_NEXT ' just ignore and continue
        Exit Function

    Case 75 ' Path/file Access Error
        err75counter = err75counter + 1
        If err75counter >= 6 Then
            MessageBox "Fehler bei Dateizugriff; " & err75counter &_
                " Versuche erfolglos, Abbruch"
            HandleError = ERROR_HANDLER_END ' abort
            Exit Function

        Else
            Print "Fehler bei Dateizugriff; versuche es nach 5 Sekunden noch " &_
                "einmal (Versuch " & err75counter & " von 6)"
            Sleep 5
            HandleError = ERROR_HANDLER_RETRY ' retry
            Exit Function
        End If
End Select
```

04 – Fehlerbehandlung – Variante C

- HandleError(), 2. Teil (wie bisher)

```
errorMessage = "Fehler #" & Err & " in Zeile " & Erl & _
    " aufgetreten: " & Chr$(10) & Error$ & Chr$(10) & Chr$(10)
errorMessage = errorMessage & "Routine " &_
    GetThreadInfo(LSI_THREAD_PROC) & Chr$(10)
errorMessage = errorMessage & "Aufrufer " &_
    GetThreadInfo(LSI_THREAD_CALLPROC) & Chr$(10)

If Not session.IsOnServer() Then ' LSI_Info is not thread-safe
    stack = LSI_Info(14)
    If IsArray(stack) Then stack = Join(stack, Chr$(10))
    errorMessage = errorMessage & "Stack " & stack
End If

MessageBox errorMessage, MB_ICONSTOP, "Fehler aufgetreten"
```

assono

04 – Fehlerbehandlung – Ausblick

- bisher nur Ausgabe für Benutzer, besser (zusätzlich)
  - speichern
    - als Dokument in der aktuellen DB oder zentral
    - in eine Datei
    - in ein anderes System (Windows Events, Syslog-Server)
  - benachrichtigen
    - per E-Mail senden (Verknüpfung oder komplette Meldung)
- mehr Informationen sammeln, unter anderem:
  - aktueller Benutzer, seine Rollen und Rechte
  - aktuelle Datenbank, Agent usw.
  - Version der Schablone/Anwendung
  - Speicherzustand, Plattenplatz usw.

assono

05 – Formelsprache nutzen in LotusScript

- einige Aufgaben gehen mit Formelsprache einfacher
- Formelsprache ausführen mit **`Evaluate()`**

```
result = Evaluate(|@Unique|)
MessageBox "Eindeutige ID: " & result(0)

result = Evaluate(|@UserRoles|)
MessageBox "Rollen des aktuellen Benutzers: " & Join(result, ", ")
```

assono

05 – Formelsprache nutzen in LotusScript

- Zugriff auf die Windows-Registry:

```
result = Evaluate(|
_productName := @RegQueryValue("HKEY_LOCAL_MACHINE";
"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion";
"ProductName");
_owner := @RegQueryValue("HKEY_LOCAL_MACHINE";
"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion";
"RegisteredOwner");
"Ihr Betriebssystem '" + _productName + "' wurde
registriert auf '" + _owner + "'"|)
MessageBox "Aus der Windows-Registry: " & Chr$(10) &
result(0)
```

assono

05 – Formelsprache nutzen in LotusScript

- **Formelsyntax prüfen**

```
formula = InputBox$("Bitte geben Sie eine Formel ein")
formula = Replace(formula, |"|, |\"|)
result = Evaluate(|@CheckFormulaSyntax("| & formula & |")|)
If result(0) = "1" Then
    MessageBox "Die Formel hat eine korrekte Syntax"
Else
    MessageBox "Die Formel hat keine korrekte Syntax: " &_
        Chr$(10) & result(0) & Chr$(10) & Chr$(10) & result(5)
End If
```

05 – Formelsprache nutzen in LotusScript

- andere gehen sehr viel einfacher und kürzer
- **Evaluate(formel, NotesDocument)**

```
message = "Gesamtgröße aller Anhänge des gewählten Dokuments: "
size = 0
Set rtItem = selectedDoc.GetFirstItem("Anhänge")
If rtItem.Type = RICHTEXT Then
    ForAll eo In rtItem.EmbeddedObjects
        If eo.Type = EMBED_ATTACHMENT Then
            size = size + eo.FileSize
        End If
    End ForAll
End If
MessageBox message & CStr(size) & " Bytes"

result = Evaluate(|@Sum(@AttachmentLengths)|, selectedDoc)
MessageBox "Gesamtgröße aller Anhänge des gewählten " &_
    " Dokuments: " & Join(result) & " Bytes"
```

assono

06 notes.ini-Einstellungen lesen

- notes.ini-Einstellungen auslesen mit
  - **`session.GetEnvironmentString(name, system)`**
  - **`session.GetEnvironmentValue(name, system)`**
- Wenn **`system = True`** ist, dann
  - Zugriff auf Einstellungen ohne $ davor
  - beschränktes LotusScript (Sicherheit!)

```
location = StrLeft(session.GetEnvironmentString("Location", _
    True) + ",", ",")

programDir = session.GetEnvironmentString("NotesProgram", True)
```

assono

06 notes.ini-Einstellungen schreiben

- notes.ini-Einstellungen schreiben mit
  - **session.SetEnvironmentVar(name, value, system)**

```
lastTimeExecuted = session.GetEnvironmentValue("IWasHere", False)
If CStr(lastTimeExecuted) = "" Then
    MessageBox "Ich war hier noch nie!"
Else
    MessageBox "Ich war hier zuletzt: " & lastTimeExecuted
End If
Call session.SetEnvironmentVar("IWasHere", Now, False)
```

07 Umgebungsvariablen lesen

- Umgebungsvariablen lesen mit
  - **Environ$(name oder index)**

```
tempDir = Environ$("TEMP")
MessageBox "Temporäres Verzeichnis: " & tempDir


computerName = Environ$("COMPUTERNAME")
MessageBox "Rechnername: " & computerName


comSpec = Environ$("ComSpec")
MessageBox "Eingabeaufforderung: " & comSpec
```

08 Dateien lesen

- Textdateien auslesen

```
Public Function GetNotesIni(numberOfRows As Long) As String
    […]
    allRows = ""
    rowNo = 0
    fileNo = FreeFile
    Open programDir & "/notes.ini" For Input As #fileNo
    Do Until EOF(fileNo) Or rowNo > numberOfRows
        rowNo = rowNo + 1
        Line Input #fileNo, row
        allRows = allRows & row & Chr$(10)
    Loop
    Close #fileNo
    GetNotesIni = allRows
End Function
```

assono

08 Dateien schreiben

- Textdateien schreiben

```
Public Sub SaveToFile(fileName As String, content As String)
    Dim fileNo As Integer


    fileNo = FreeFile
    Open fileName For Output Access Write Lock Read Write _
        As #fileNo Charset = "UTF-8"
    Print #fileNo, content
    Close #fileNo
End Sub
```

assono

08 Dateien schreiben

- und jetzt beides zusammen

```
notesIni = GetNotesIni(2000)

tempDir = Environ$("TEMP")

fileNames = uiws.SaveFileDialog(False, _
    "Wo soll die Kopie der notes.ini gespeichert werden?",_
    "Konfigurationsdateien|*.ini|Textdateien|" &_
    "*.txt|Alle Dateien|*", tempDir, _
    "notes.ini-Kopie-" & CreateTimestamp() & ".txt")
If IsEmpty(fileNames) Then Exit Sub ' user cancelled

Call SaveToFile(fileNames(0), notesIni)
```

assono

09 Dynamisches LotusScript ausführen

- dynamisch = LotusScript-Code, der erst zur Laufzeit compiliert und sofort ausgeführt wird
- **`returnCode = Execute(script)`**

```
Const LS_CODE$ = |MessageBox "Aktueller Zeitpunkt: " & Now
MessageBox "Aktuelle Datenbank: " & currentDB.Title|

Dim returnCode As Integer

returnCode = Execute(LS_CODE)
If returnCode <> 0 Then
   MessageBox "Ausführung war nicht erfolgreich, Rückgabewert: " &_
      returnCode
End If
```

- Zugriff nur auf globale Variablen und Routinen möglich (Variable **`currentDB`** im Beispiel)

assono

09 Dynamisches LotusScript ausführen

- dynamisches Laden von Script-Bibliotheken,
  z. B. um zyklische Abhängigkeiten zu vermeiden

```
Const LS_CODE$ = |Use "08 Dateien"
MessageBox GetNotesIni(5)|

Dim returnCode As Integer

returnCode = Execute(LS_CODE)
    If returnCode <> 0 Then
    MessageBox "Ausführung war nicht erfolgreich, Rückgabewert: " &_
        returnCode
End If
```

- Ladezeit wächst leicht exponentiell, wenn sehr viele Script-Bibliotheken „normal“ geladen werden

assono

09 Dynamisches LotusScript ausführen

- Code kann sogar in Dokumenten gespeichert werden:

```
lsCode = Join(selectedDoc.GetItemValue("LotusScriptCode"), Chr$(10))

returnCode = Execute(lsCode)
If returnCode = 0 Then
    MessageBox "Ausführung war erfolgreich„
Else
    MessageBox "Ausführung war nicht erfolgreich, Rückgabewert: " &_
        returnCode
End If
```

- Integration/Entkopplung von anderen Systemen
- Konfiguration Entwicklung/Test/produktive Umgebung

assono

10 Dateianhänge exportieren

- Anhänge können in RichText-Item oder Dokument selbst (V2-Stil) gespeichert sein

```
ReDim eos(0 To 0) As NotesEmbeddedObject
Set eos(0) = Nothing

If itemName = "" Then
    eos = doc.EmbeddedObjects
Else
    If doc.HasItem(itemName) Then
        Set rtItem = doc.GetFirstItem(itemName)
        If Not rtItem Is Nothing Then
            If rtItem.Type = RICHTEXT Then
                eos = rtItem.EmbeddedObjects
            End If
        End If
    End If
End If
```

10 Dateianhänge exportieren

- Der eigentliche Export: **`eo.ExtractFile(fileName)`**, der ursprüngliche Dateiname steht in **`eo.Source`**

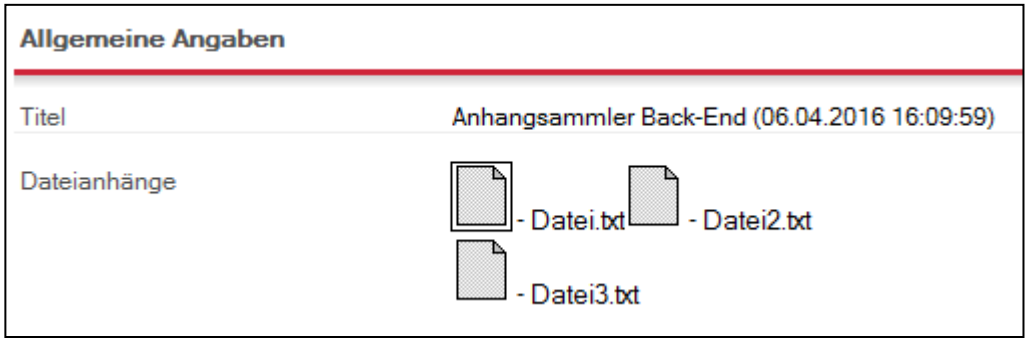
```
ReDim fileNames(0 To 0)
fileNames(0) = ""

If IsArray(eos) Then
    If Not eos(LBound(eos)) Is Nothing Then
        ReDim fileNames(LBound(eos) To Bound(eos)) As String

        For i = LBound(eos) To UBound(eos)
            Set eo = eos(i)
            fileName = filePath & "\" & eo.Source
            Call eo.ExtractFile(fileName)
            fileNames(i) = fileName
        Next
    End If
End If
```

10 Dateianhänge importieren - Back-End

- RichTextItem wie gewohnt erzeugen und dann:

```
Dim i As Integer

For i = LBound(fileNames) To UBound(fileNames)
    Call rtItem.Embedobject(EMBED_ATTACHMENT, "", fileNames(i))
Next
```

- Das Ergebnis sieht dann so aus:



Es gibt keine „schönen“ Icons.

assono

10 Dateianhänge importieren - Front-End

- Eine Methode für NotesUIDocument (Front-End) um Dateien „schön“ anzuhängen gibt es nicht.
- @Command([EditInsertFileAttachment]; _fileName)
- Aber @Commands funktionieren nicht mit Evaluate…
- NotesAgent.Run mit Formelsprache-Agent geht auch nicht…
- Es geht mit zwei Agenten (LotusScript und Formelsprache), die aber beide nacheinander in Formelsprache gestartet werden müssen. (Attaching a file in the front-end from LotusScript)
- Oder man macht es wie folgt:

assono

10 Dateianhänge importieren - Front-End

- LotusScript im Back-End:

```
Set newDoc = currentDB.CreateDocument
Call newDoc.ReplaceItemValue("Form", "Dateianhänge")
Call newDoc.ReplaceItemValue("Titel", titel)
Call newDoc.ReplaceItemValue("ZuImportierendeDateien", fileNames)
Set uidoc = uiws.EditDocument(True, newDoc)
Call uidoc.Save()
Call uidoc.Close(True)
```

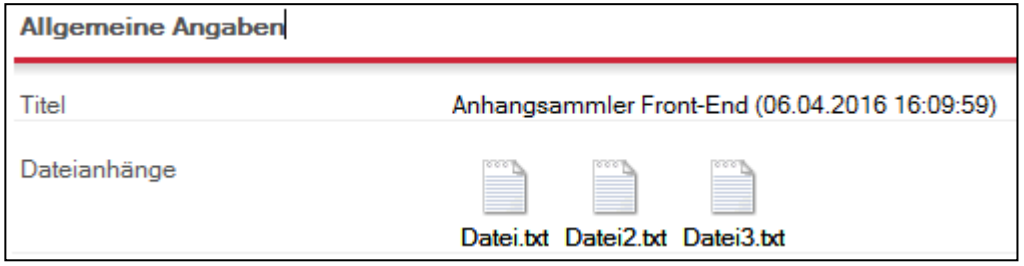
10 Dateianhänge importieren - Front-End

- Formelsprache im PostOpen-Ereignis der Maske:

```
@If(
    ZuImportierendeDateien = "";
    @Return("nothing to import");
    "continue with import„
);

@Command([EditGotoField]; "Anhänge");

_fileNames := ZuImportierendeDateien;
_numberOfFiles := @Elements(_fileNames);
@For(
    i := 0;   i <= _numberOfFiles;   i := i + 1;

    @Command([EditInsertFileAttachment]; _fileNames[i]; "1"; "1")
);

FIELD ZuImportierendeDateien := "";

"import done"
```

10 Dateianhänge importieren - Front-End

- Und das Ergebnis:



11 Externe Programme

- Externe Programme aufrufen mit
  - **returnCode = Shell(program, windowStyle)**

```
comSpec = Environ$("ComSpec")
returnCode = Shell(|"| & comSpec & |" /S /K PUSHD "%TEMP%"|,_
    SHELL_NORMAL_FOCUS)
If returnCode <> 33 Then
    MessageBox "Irgendein Problem ist aufgetreten?!„
End If
```

- Das Programm wird gestartet und LotusScript sofort weiter ausgeführt.
- kein Zugriff auf das Ergebnis des Programmlaufs
- Wie kann man Dateien öffnen und danach löschen?

11 Externe Programme – Batch-Dateien

- Idee: Batch-Datei dynamisch erzeugen und starten
- Batch-Datei startet dann passende Programm, öffnet die Datei, wartet auf das Ausführungsende und löscht dann die Datei.

```
fileName = tempDir & "\" & oe.Source

Call oe.ExtractFile(fileName)

Call OpenAndDeleteFile(fileName, Environ$("USERPROFILE"))
```

- Batch-Dateien können nur indirekt über cmd.exe gestartet werden.

11 Externe Programme – Batch-Dateien

- **Public Sub OpenAndDeleteFile(fileName, folder)**

```
tempDir = Environ$("TEMP")
batchFileName = tempDir & "/OpenAndDelete-" &_
    CreateTimestamp() & ".cmd"
fileName = Replace(fileName, "%", "%%")
fileNo = FreeFile()

Open batchFileName For Output Access Write As #fileNo
Print  #fileNo, |%%Echo Off
START "Open File" /D"| & folder & |" /WAIT "| & fileName & |"
DEL /F "| & fileName & |"
DEL /F "| & batchFileName & |"|
Close #fileNo

comSpec = Environ$("ComSpec")
returnCode = Shell(|"| & comSpec & |" /C "| & batchFileName  &_
    |"|, SHELL_MIN_NO_FOCUS)
```

assono

12 ODBC-Datenbankzugriff

- 2 Möglichkeiten:
  - Lotus Connector for ODBC
  - LotusScript Data Object (LS:DO)
- Ich verwende LS:DO, weil mir einfacher erscheint.
- Es gibt nur drei Klassen:
  - ODBCConnection
  - ODBCQuery
  - ODBCResultSet

assono

12 ODBC-Datenbankzugriff – LS:DO

- Verbindung aufbauen:

```
On Error Resume Next ' to deactivate the standard error handling
Call connection.ConnectTo(dsn, userName, password)
On Error GoTo 0

If connection.GetError <> DBstsSUCCESS Then
    LogODBCError connection, "Connect - connection.ConnectTo„
End If

' not connected => ODBC error has occurred
If Not(connection.IsConnected) Then

[…]
```

12 ODBC-Datenbankzugriff – LS:DO

- Verbindung beenden:

```
If Not connection.AutoCommit Then
If Not connection.RollbackTransactions() Then
    LogODBCError connection, "Disconnect - RollbackTransactions"
End If
End If

If Not connection.Disconnect Then
    LogODBCError connection, "Disconnect - Disconnect"
End If
```

12 ODBC-Datenbankzugriff – LS:DO

- minimale Fehlerprotokollierung:

```
message = "ODBC-Fehler: " & message & Chr$(10)

message = message & |ODBC-Fehler #| & CStr(odbcObject.GetError)_
& | aufgetreten: | & CStr(odbcObject.GetErrorMessage) & Chr$(10)

message = message & |Erweiterte ODBC-Fehlermeldung: | &_
CStr(odbcObject.GetExtendedErrorMessage) & Chr$(10)

MessageBox message, MB_ICONSTOP, "ODBC-Fehler"
```

assono

12 ODBC-Datenbankzugriff – LS:DO

- Tabelle lesen:

```
Set query = New ODBCQuery()
Set query.Connection = connection
query.SQL = "SELECT * FROM " & tableName

Set resultSet = New ODBCResultSet()
Set resultSet.Query = query
resultSet.MaxRows = 100
resultSet.FetchBatchSize = 100
resultSet.CacheLimit = 100

Call resultSet.Execute()
If resultSet.GetError() <> DBstsSUCCESS Then
    LogODBCError resultSet, "resultSet.Execute„
    Exit Sub
End If
```

12 ODBC-Datenbankzugriff – LS:DO

- Tabelle lesen (forts.):

```
ReDim result(0 To resultSet.NumRows - 1)
rowNumber = 0
Do
    Call resultSet.NextRow

    ReDim row(0 To resultSet.NumColumns - 1)
    For columnNumber = 0 To resultSet.NumColumns - 1
        row(columnNumber) = resultSet.GetValue(columnNumber + 1)
    Next

    If rowNumber > UBound(result) Then
        ReDim result(0 To UBound(result) + 1)
    End If

    result(rowNumber) = row
    rowNumber = rowNumber + 1
Loop Until resultSet.IsEndOfData
```

12 ODBC-Datenbankzugriff – LS:DO

- Zeile anhängen

```
Set query = New ODBCQuery()
Set query.Connection = connection
query.SQL = "SELECT * FROM " & table & " WHERE 1=0" ' empty result

Set resultSet = New ODBCResultSet()
Set resultSet.Query = query
resultSet.MaxRows = 1
resultSet.FetchBatchSize = 1
resultSet.CacheLimit = 100

Call resultSet.Execute()
If resultSet.GetError() <> DBstsSUCCESS Then
    LogODBCError resultSet, "resultSet.Execute"
    Exit Sub
End If
```

12 ODBC-Datenbankzugriff – LS:DO

- Zeile anhängen (forts.)

```
Call resultSet.AddRow
If resultSet.GetError() <> DBstsSUCCESS Then
[…]

offset = 1 - LBound(data) ' columns are 1 based
For i = LBound(data) To UBound(data)
    Call resultSet.SetValue(i + offset, data(i))
Next

Call resultSet.UpdateRow()
If resultSet.GetError() <> DBstsSUCCESS Then
[…]

resultSet.Close(DB_CLOSE)
If resultSet.GetError() <> DBstsSUCCESS Then
[…]
```

assono

13 Excel fernsteuern

- Excel über COM-Schnittstelle starten und steuern
- zunächst einmal eine Referenz auf Excel holen
- dabei wird Excel ggf. gestartet

```
Function ConnectToOfficeApp(className As String) As Variant
    Set ConnectToOfficeApp = Nothing
    On Error ErrOLECantCreate GoTo oleCantCreateHandler
    Set ConnectToOfficeApp = GetObject(,className)
    Exit Function
oleCantCreateHandler:
    Set ConnectToOfficeApp = CreateObject(className)
    Resume Next
End Function
```

assono

13 Excel fernsteuern

- Vorbereitungen

```
Set xlApp = ConnectToOfficeApp("Excel.Application")
xlApp.Visible = False


Call xlApp.Workbooks.Add
xlApp.ActiveSheet.EnableCalculation = False
xlApp.ActiveSheet.Name = "notes.ini"
```

13 Excel fernsteuern

- Überschriften und in Zellen schreiben

```
Call WriteToCell(xlApp, 1, 1, "Variable")
Call WriteToCell(xlApp, 2, 1, "Wert")

Call SetBold(xlApp, 1, 1, 2, 1)

Call SetBorders(xlApp, 1, 1, 2, 1, xlMedium, xlThin, True, True)
```

```
Public Sub WriteToCell(xlApp, column, row, content)

If IsArray(content) Then
    xlApp.ActiveSheet.Cells(row, column) = Join(content,Chr$(10))
Else
    xlApp.ActiveSheet.Cells(row, column) = content
End If
End Sub
```

assono

13 Excel fernsteuern

- Formatierungen

```
Public Sub SetBold(xlApp, fromColumn, fromRow, toColumn, toRow)
Call SelectRange(xlApp, fromColumn, fromRow, toColumn, toRow)
xlApp.Selection.Font.Bold = True
End Sub
```

```
Public Sub SetBorders(xlApp, fromColumn, fromRow, toColumn,toRow,
outerBorderWith, innerBorderWith, innerHorizontals, innerVerticals)
    With xlApp.Selection
        .Borders(xlEdgeLeft).LineStyle = xlContinuous
        .Borders(xlEdgeLeft).Weight = outerBorderWith
        .Borders(xlEdgeTop).LineStyle = xlContinuous
        .Borders(xlEdgeTop).Weight = outerBorderWith
        […]
    End With
End Sub
```

assono

13 Excel fernsteuern

- Daten schreiben

```
currentRowNumber = FIRST_DATA_ROW
For i = LBound(notesIniRows) To UBound(notesIniRows)
    notesIniRow = notesIniRows(i)
    variable = StrLeft$(notesIniRow, "=")
    value = StrRight$(notesIniRow, "=")

    Call WriteToCell(xlApp, 1, currentRowNumber, variable)
    Call WriteToCell(xlApp, 2, currentRowNumber, value)

    currentRowNumber = currentRowNumber + 1
Next
```

13 Excel fernsteuern

- Rest formatieren und abschließen

```
Call SetBorders(xlApp, 1, FIRST_DATA_ROW, 2, currentRowNumber - 1,_
    xlMedium, xlThin, True, True)

Call SetAutoFilter(xlApp, 1, 1, 2, "")

Call SetColumnsAutoFit(xlApp, 1, 2)

Call SetHorizontalAlignment(xlApp, 1, 1, 2, currentRowNumber - 1,_
    xlLeft)

Call SelectRange(xlApp, 1, 1, "", "")


xlApp.ActiveSheet.EnableCalculation = True

xlApp.Visible = True
```

13 Excel fernsteuern

- Formatierungen

```
Public Sub SetAutoFilter(xlApp, fromColumn, fromRow, toColumn, toRow)
    Call SelectRange(xlApp, fromColumn, fromRow, toColumn, toRow)
    xlApp.Selection.AutoFilter
End Sub
```

```
Public Sub SetColumnsAutoFit(xlApp, fromColumn, toColumn)
    xlApp.ActiveSheet.Rows(fromColumn & ":" & toColumn)._
        EntireColumn.AutoFit
End Sub
```

```
Public Sub SetHorizontalAlignment(xlApp, fromColumn, fromRow,_
    toColumn, toRow, alignment)
    Call SelectRange(xlApp, fromColumn, fromRow, toColumn, toRow)
    xlApp.Selection.HorizontalAlignment = alignment
End Sub
```

14 PDF-Export

- Dokumente in PDF-Dateien drucken – automatisch
- benutze PDFCreator 1.7.3
  - Versionen >= 2 haben neues Objektmodell

```
Set PDFCreator = CreateObject("PDFCreator.clsPDFCreator")

If Not PDFCreator.cStart("/NoProcessingAtStartup", True) Then _
    Error 4000, "PDFCreator-Drucker ist nicht installiert"
PDFCreator.cClearCache

PDFCreator.cOption("UseAutosave") = 1
PDFCreator.cOption("UseAutosaveDirectory") = 1
PDFCreator.cOption("AutosaveDirectory") = filePath & "/"
PDFCreator.cOption("AutosaveFilename") = fileName
PDFCreator.cOption("AutosaveFormat") = 0' 0=PDF, 1=PNG, 2=JPG, …
PDFCreator.cOption("OptionsVisible") = 0
PDFCreator.cOption("PDFOptimize") = 1
```

14 PDF-Export

- nach der langen Vorbereitung…

```
On Error 4407 Resume Next
    ' if there are no sections, this throws an 4407 error
    Call uidoc.ExpandAllSections()
On Error GoTo 0

Set uidoc = uiws.EditDocument(False, doc, True)
Call uidoc.Print(1, , , False, "PDFCreator")
Call uidoc.Close(True)

PDFCreator.cPrinterStop = False

' wait until the file shows up before continuing
Do
    Yield
    Sleep 0.1
Loop Until ExistsFile(filePath & "/" & fileName)
```

15 InViewEdit

- Direkt in der Ansicht Dokumente bearbeiten und neue erstellen
- einfacher Fall: Icon-Spalte anklicken
- 1. Beispiel: Wechsel zwischen gelesen und ungelesen
- 2. Beispiel: Bewertung mit Sternen

assono

15 InViewEdit

- Im Kern: NotesUIView.InViewEdit-Ereignis

```
Sub InViewEdit(Source As NotesUIView, RequestType As Integer, _
ColProgName As Variant, ColumnValue As Variant, Continue As Variant)

    caret = Source.CaretNoteID
    If caret = "0" Then Exit Sub

    Set currentDB = Source.View.Parent
    Set currentDoc = currentDB.GetDocumentByID(caret)
    If currentDoc Is Nothing Then Exit Sub

    Select Case RequestType
        Case QUERY_REQUEST ' = 1; ignore
        Case VALIDATE_REQUEST ' =2; ignore
        Case SAVE_REQUEST ' = 3
            […]
        Case NEWENTRY_REQUEST ' = 4; ignore
    End Select
End Sub
```

assono

15 InViewEdit

- Status: gelesen/ungelesen

```
Case SAVE_REQUEST ' = 3
    columnName = ColProgName(0)
    If columnName = "Status" Then
        currentStatus = currentDoc.GetItemValue("Status")(0)
        If currentStatus = "gelesen" Then
            Call currentDoc.ReplaceItemValue("Status", "ungelesen")
        Else
            Call currentDoc.ReplaceItemValue("Status", "gelesen")
        End If
    ElseIf […]
```

assono

15 InViewEdit

- Bewertung: 0 bis 5 Sterne

```
Case SAVE_REQUEST ' = 3
        […]
    ElseIf Left$(columnName, 6) = "$Stern" Then
        currentStars = currentDoc.GetItemValue("Bewertung")(0)
        starNo = CInt(StrRight(columnName, "$Stern"))
        If starNo <> currentStars Then
            Call currentDoc.ReplaceItemValue("Bewertung", starNo)
        Else ' starNo = currentStars
            Call currentDoc.ReplaceItemValue("Bewertung", starNo - 1)
        End If
    End If
    Call currentDoc.Save(True, True, True)
```

16 Agenten starten – Server-Agent

- Agent auf Server starten mit Parameterübergaben in beide Richtungen:

```
Set parameterDoc = currentDB.CreateDocument
Call parameterDoc.ReplaceItemValue("Form", "ParameterDokument")
Call parameterDoc.ReplaceItemValue("Wert", _
    InputBox$("Geben Sie einen Wert ein:"))

Call parameterDoc.Save(True, True)
parameterDocID = parameterDoc.NoteID

Set agent = currentDB.GetAgent("16a Server-Agent")
Call agent.RunOnServer(parameterDocID)

Delete parameterDoc
Set parameterDoc = currentDB.GetDocumentByID(parameterDocID)

MessageBox "Der Agent ist gelaufen; Rückgabe:" & Chr$(10) &_
    parameterDoc.GetItemValue("Rückgabe")(0)
```

16 Agenten starten – Server-Agent

- auf der anderen Seite:

```
Set currentAgent = session.CurrentAgent

parameterDocID = currentAgent.ParameterDocID
Set parameterDoc = currentDB.GetDocumentByID(parameterDocID)

If parameterDoc Is Nothing Then
    Print "Das Parameter-Dokument wurde nicht gefunden."

Else
    value = parameterDoc.GetItemValue("Wert")(0)
    Print "Es wurde folgender Wert übergeben: '" & value & "'"

    Call parameterDoc.ReplaceItemValue("Rückgabe",_
        "Es ist jetzt: " & Now)
    Call parameterDoc.Save(True, True)
End If
```

assono

16 Agenten starten – Agent mit „Kontext“

- Agent mit Parameterübergaben in beide Richtungen, aber diesmal ohne zu speichern:

```
Set parameterDoc = currentDB.CreateDocument
Call parameterDoc.ReplaceItemValue("Form", "ParameterDokument")
Call parameterDoc.ReplaceItemValue("Wert",_
    InputBox$("Geben Sie einen Wert ein:"))

Set agent = currentDB.GetAgent("16b Agent mit Kontext")
Call agent.RunWithDocumentContext(parameterDoc)

If parameterDoc.HasItem("Rückgabe") Then
    MessageBox "Der Agent ist gelaufen; Rückgabe:" & Chr$(10) &_
        parameterDoc.GetItemValue("Rückgabe")(0)
End If
```

assono

16 Agenten starten – Agent mit „Kontext“

- auf der anderen Seite auch viel kürzer:

```
Set parameterDoc = session.DocumentContext

If parameterDoc.HasItem("Wert") Then
    value = parameterDoc.GetItemValue("Wert")(0)
    Print "Es wurde folgender Wert übergeben: '" & value & "'"
End If

Call parameterDoc.ReplaceItemValue("Rückgabe", _
    "Es ist jetzt: " & Now)
```

- keine Löschrechte notwendig!
- sprachübergreifend: Java und XPages (SSJS)

17 NotesAdministrationProzess

- Auftrag an AdminP erstellen

```
Set adminDB = New NotesDatabase(server, "admin4.nsf")

If Not adminDB Is Nothing Then
    Set adminP = session.CreateAdministrationProcess(server)

    noteID = adminP.SignDatabaseWithServerID(server,_
        currentDB.FilePath, False)

    If noteID <> "" Then
        Call uiws.EditDocument(False, _
            adminDB.GetDocumentByID(noteID))
    End If

End If
```

assono

18 SendConsoleCommand

- Befehl direkt auf der Serverkonsole ausführen lassen:

```
dropCommand = "drop " & filePath
consoleReturn = session.SendConsoleCommand(server, dropCommand)

compactCommand = "load compact -B " & filePath
consoleReturn = session.SendConsoleCommand(server, compactCommand)
```

- Unterzeichner braucht Rechte auf dem Server:
  - Server-Dokument
  - Reiter „Sicherheit“
  - Abschnitt „Administratoren“
  - „Administratoren mit voller Remotekonsolen-Berechtigung“

assono

19 LotusScript-Klassen

- schon immer verwendet: NotesDatabase, NotesView, NotesDocument, …
- Man kann selbst neue Klassen erstellen.
- **viele** Vorteile (gegenüber Prozeduren/Funktionen):
  - hohe Wiederverwendung ohne bestehenden Code zu kopieren oder zu verändern
  - hohe Flexibilität
  - exzellente Erweiterbarkeit
  - leichtere Wartbarkeit
  - geringere Fehlerrisiken
  - usw.

19 LotusScript-Klassen

- Beispiel: E-Mails generieren und verschicken
  - `SendMail(sendTo, subject, message)`
  - `SendMail(sendTo, CopyTo, BCCTo, subject, message)`
  - `SendMail(sendTo, CopyTo, BCCTo, principal, subject, message)`
  - `SendMail(sendTo, CopyTo, BCCTo, principal, subject, message, docLink)`
- formatierter (Rich-)Text, Dateianhänge, zusätzliche Items (z.B. $AutoForward), hohe Dringlichkeit, usw.
- Und in Wirklichkeit müssen alle Prozeduren unterschiedliche Namen haben!

assono

19 LotusScript-Klassen – Klasse EMail

- Klasse: **`Public Class EMail`**
- Attribute:
  - **`sender`**
  - **`principal`**
  - **`sendTo`**
  - **`ccTo`**
  - **`bccTo`**
  - **`subject`**
  - **`additionalMailItems`**
  - **`bodyRTItem`**
  - **`mailDoc`**
  - **`mailDB`**
- Methoden:
  - **`AppendText`**
  - **`AddNewLine`**
  - **`AppendDocLink`**
  - **`CreateStyle`**
  - **`AddStyle`**
  - **`SetStyle`**
  - Getter und Setter

19 LotusScript-Klassen – Verwendung Klasse EMail

- Verwendung:

```
Dim mail As EMail

Set mail = New EMail()
Call mail.SetSendTo("tbahn@assono.de")
Call mail.SetSubject("1. Test")
Call mail.AppendText("1. Test")
Call mail.Send
```

- oder kompakter mit „Chaining“:

```
Set mail = New EMail()

Call mail.SetSendTo("tbahn@assono.de").SetSubject("2. Test")._
    AppendText("2. Test").Send
```

assono

19 LotusScript-Klassen – Verwendung Klasse EMail

- äußerst flexibel nutzbar:

```
Set mail = New EMail()
Call mail.SetSendTo("tbahn@assono.de").SetSubject("3. Test")
Call mail.SetStyle("headline").AppendText("3. Test")._
    SetStyle("standard").AddNewLine(2)
Call mail.AppendText("Die direkte Verwendung der EMail-Klasse " &_
    "erlaubt komplexe E-Mails").AddNewLine(2)
Call mail.AppendText("Verknüpfung zur " & currentDB.Title &_
    "-Datenbank: ").AppendDocLink(currentDB, currentDB.Title, "")_
    .AddNewLine(1)
Call mail.AddAdditionalMailItem("AktuellerBenutzer", session.UserName)
Call mail.AddAdditionalMailItem("ReplyTo", "tbahn@assono.de")
Call mail.Send()
```

assono

19 LotusScript-Klassen – Klasse DirectEMail

- erweiterbar:

```
Public Class DirectEMail As EMail

    Private directSend As Boolean
    Public Function GetDirectSend As Boolean
    Public Function SetDirectSend(directSend As Boolean) As DirectEMail

    Public Function SetSender(sender As String) As DirectEMail

    Public Sub New()

    Private Function GetMailBoxDB() As NotesDatabase

    Private Sub SendEMail()

End Class
```

- vorhandener Code bleibt unverändert (kein Testen!)

assono

19 LotusScript-Klassen – Verwendung DirectEMail

- Verwendung:

```
Set directMail = New DirectEMail()
Call directMail.SetDirectSend(True).SetSender(currentDB.Title & "-DB")
Call directMail.SetSendTo("tbahn@assono.de").SetSubject("4. Test")
Call directMail.AppendText("gesendet aus Datenbank " &_
    currentDB.Title & " von Server " &_
    session.CreateName(currentDB.Server).Abbreviated).AddNewLine(1)
Call directMail.AppendText("'echter' Absender: " &_
    session.CreateName(session.UserName).Abbreviated)
Call directMail.Send()
```

assono

20 NotesDbDirectory – DirectoryWalker

- erlaubt das Durchlaufen aller Datenbanken und Schablonen auf einem Server (oder lokal)

```
Public Class DirectoryWalker

    […]

    Public Sub Start()

        Set dbDir = New NotesDbDirectory(Me.serverName)
        Set db = dbdir.GetFirstDatabase(Me.dbTypes)
        Do While Not(db Is Nothing)
            Call VisitDB(db)
            Set db = dbDir.GetNextDatabase
        Loop
    End Sub

    Private Sub VisitDB(db As NotesDatabase)
    End Sub

End Class
```

assono

20 NotesDbDirectory – DirectoryPrintWalker

- einfach ausgeben:

```
Public Class DirectoryPrintWalker As DirectoryWalker

    Sub New(serverName As String, dbTypes As Integer),_
        DirectoryWalker(serverName, dbTypes)

    End Sub ' DirectoryPrintWalker.New

    Private Sub VisitDB(db As NotesDatabase)

        Print db.Title, db.FilePath

    End Sub ' DirectoryPrintWalker.VisitDB

End Class
```

assono

20 NotesDbDirectory – DirectoryFileDumpWalker

- oder in eine Datei schreiben:

```
Public Class DirectoryPrintWalker As DirectoryWalker

    Sub New(serverName As String, dbTypes As Integer, _
        dumpFileName As String), DirectoryWalker(serverName, dbTypes)

    Public Sub Start()
        fileNumber = FreeFile
        Open dumpFileName For Output As #fileNumber Charset = "UTF-8"

        Call DirectoryWalker..Start()

        Close #fileNumber
    End Sub

    Private Sub VisitDB(db As NotesDatabase)
        Print #fileNumber, db.Title & "" & db.FilePath & "" & db.Size
    End Sub

End Class
```

assono

20 NotesDbDirectory – DirectoryFileDumpWalker

- Verwendung:

```
Set dirWalker = New DirectoryPrintWalker(currentDB.Server, _
    TEMPLATE_CANDIDATE)
Call dirWalker.Start()


fileNames = uiws.SaveFileDialog(False, "Name der Datei", _
    "Textdateien|*.txt|CSV-Dateien|*.csv|Alle Dateien|*.*", "", "")
If IsEmpty(fileNames) Then Exit Sub ' user cancelled

Set dirFileDumpWalker = New DirectoryFileDumpWalker(_
    currentDB.Server, TEMPLATE_CANDIDATE, fileNames(0))
Call dirFileDumpWalker.Start()
```

assono

21 NotesACL – DirectoryACLWalker

- Klassen NotesACL und NotesACLEntry erlauben auslesen und ändern der ACL:

```
Public Class DirectoryACLWalker As DirectoryWalker

Private Sub VisitDB(db As NotesDatabase)

    Set dbACL = db.Acl
    allRoles = dbACL.Roles

    If Me.namesToBeRemoved(0) <> "" Then
        ForAll nameToBeRemoved In Me.namesToBeRemoved
            Set aclEntry = dbACL.GetEntry(nameToBeRemoved)
            If Not aclEntry Is Nothing Then
                Call aclEntry.Remove()
            End If
        End ForAll
    End If
[…]
```

assono

21 NotesACL – DirectoryACLWalker

- Fortsetzung **VisitDB**:

```
[…]
    If Me.namesToBeAdded(0) <> "" Then
        ForAll nameToBeAdded In Me.namesToBeAdded
            Set aclEntry = dbACL.GetEntry(nameToBeAdded)
            If aclEntry Is Nothing Then
                Set aclEntry = dbACL.CreateACLEntry(nameToBeAdded, 6)
                Call AssertFullAccess(aclEntry, allRoles)
            Else
                Call AssertFullAccess(aclEntry, allRoles)
            End If
        End ForAll
    End If
    Call dbACL.Save()
End Sub
```

assono

21 NotesACL – DirectoryACLWalker

- Eigenschaften eines ACL-Eintrags:

```
Sub AssertFullAccess(aclEntry As NotesACLEntry, allRoles As Variant)
    aclEntry.Level = ACLLEVEL_MANAGER

    aclEntry.CanCreateDocuments = True
    aclEntry.CanDeleteDocuments = True
    aclEntry.CanCreatePersonalAgent = True
    aclEntry.CanCreatePersonalFolder = True
    aclEntry.CanCreateSharedFolder = True
    aclEntry.CanCreateLSOrJavaAgent = True
    aclEntry.IsPublicReader = True
    aclEntry.IsPublicWriter = True
    aclEntry.CanReplicateOrCopyDocuments = True

    ForAll role In allRoles
        Call aclEntry.EnableRole(role)
    End ForAll
End Sub
```

assono

21 NotesACL – DirectoryACLWalker

- Verwendung:

```
Dim dirACLWalker As DirectoryACLWalker

Set dirACLWalker = New DirectoryACLWalker(currentDB.Server, _
    TEMPLATE_CANDIDATE, "Demos\*")

Call dirACLWalker.AppendNameToBeAdded("LocalDomainServers")
Call dirACLWalker.AppendNameToBeRemoved("CN=Rainer Zufall/O=assono")

Call dirACLWalker.Start()
```

assono

22 Windows-API – Zwischenablage lesen und setzen

- In Formelsprache möglich, aber in LotusScript?
- Es geht unter Windows mit der Windows-API
- Verwendung:

```
Dim clipboardContent As String

clipboardContent = GetClipboardText(CF_TEXT)
MessageBox "Aktueller Inhalt der Zwischenablage: " &_
    clipboardContent

Call SetClipboardText("Es ist jetzt: " & Now)

clipboardContent = GetClipboardText(CF_TEXT)
MessageBox "Neuer Inhalt der Zwischenablage: " &_
    clipboardContent
```

assono

22 Windows-API – Funktionsdeklaration

- **GetClipboardText** hyperreduziert:

```
Public Function GetClipboardText(cbFormat As Integer) As String

    OpenClipboard(0&)

    handleForBuffer = GetClipboardData(cbFormat)
    pointerToBuffer = GlobalLock(handleForBuffer)

    If pointerToBuffer <> 0 Then
        clipboardData = Space(lstrlen(ByVal pointerToBuffer))
        lstrcpy(clipboardData, pointerToBuffer)
        positionNULL = InStr(1, clipboardData, Chr$(0), 0)
        If positionNULL > 0 Then
            GetClipboardText = Mid(clipboardData, 1, positionNULL - 1)
        Else
            GetClipboardText = clipboardData
        End If
        GlobalUnlock(handleForBuffer)
    End If
    CloseClipboard
End Function
```

assono

22 Windows-API – Funktionsdeklaration

- Deklarieren der Funktionen vor Benutzung:

```
Declare Function GetLastError Lib "kernel32" () As Long

Declare Function FormatMessage Lib "kernel32" Alias
"FormatMessageA" (ByVal dwFlags As Long, lpSource As Any, ByVal
dwMessageId As Long, ByVal dwLanguageId As Long, ByVal lpBuffer
As String, ByVal nSize As Long, Arguments As Long) As Long

Declare Function OpenClipboard Lib "User32"(ByVal hWnd As Long)
As Long
```

- Quelle für Deklarationen: Suche nach Visual Basic

assono

22 Windows-API – Fehlerhandling

- Typisch für Low-Level-APIs:
  - manuell das Ergebnis eines Aufrufs testen
  - Fehlerinformationen ermitteln

```
If OpenClipboard(0&) = 0 Then ' 0 means error
    errorNr = GetLastWinAPIError()
    Error errorNr, GetLastWinAPIErrorText() & " (" &_
        CStr(errorNr) & ")„
End If
```

assono

22 Windows-API – `GetLastWinAPIError`

- Hilfsfunktionen für Fehler **`GetLastWinAPIError`**:

```
Private Function GetLastWinAPIError As Long

    GetLastWinAPIError = GetLastError()
End Function
```

assono

22 Windows-API – GetLastWinAPIErrorText

- Hilfsfunktionen für Fehler: **GetLastWinAPIErrorText**

```
Private Function GetLastWinAPIErrorText As String

    Dim buffer As String * 256
    Dim valueLength As Long

    valueLength = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM Or _
        FORMAT_MESSAGE_IGNORE_INSERTS, 0&, GetLastError(),_
        LANG_NEUTRAL Or (SUBLANG_DEFAULT * 1024) , buffer, 255, 0&)

    If valueLength = 0 Then
        GetLastWinAPIErrorText = "Fehler aufgetreten bei der " &_
            Ermittlung der Fehlernachricht."
    Else
        GetLastWinAPIErrorText = Left$(buffer, valueLength)
    End If
End Function
```

assono

23 C-API

- Viele Funktionen von Notes und Domino sind nicht in direkt LotusScript nutzbar – nur in der C-API.
- Wer will schon C-Programme schreiben, DLLs erstellen und per Software-Verteilung an die Rechner bringen (oder schlimmer noch: ohne Software-Verteilung)?
- viele C-Funktionen aus LotusScript heraus aufrufbar
- 2 Beispiele
  - „Ping" auf Domino-Server
  - Temporäres Verzeichnis ermitteln (von Notes/Domino, nicht vom Betriebssystem)

```
Print "Ping " & server & ": " & GetServerLatency(server) & " ms"

Print "Aktuelles Temp-Verzeichnis: " & GetNotesTempDirectory()
```

assono

23 C-API – GetServerLatency

- **GetServerLatency**

```
Public Function GetServerLatency(serverName As String) As Long
    Call NSFGetServerLatency(_
        session.CreateName(serverName).Canonical, TIMEOUT, _
        msFromClient, msFromServer, serverVersion)
    GetServerLatency = msToServer + msFromServer
End Function
```

assono

23 C-API – NSFGetServerLatency

- **NSFGetServerLatency**

```
Public Function NSFGetServerLatency(ByVal serverName As String, _
    ByVal timeout As Long, msToServer As Long, msFromServer As Long,_
    serverVersion As Integer) As Integer

    Select Case session.Platform
        Case PLATFORM_W32
            returnCode = W32_NSFGetServerLatency(serverName, timeout, _
                msToServer, msFromServer, serverVersion)
[…]
    End Select

    If returnCode <> 0 Then ' failure
        Print "Fehler " & returnCode & " in NSFGetServerLatency': " &_
            Chr$(10) & GetCAPIErrorMsg(returnCode)
    End If
    NSFGetServerLatency = returnCode
End Function
```

assono

23 C-API – W32_NSFGetServerLatency

- **W32_NSFGetServerLatency**
- Schnittstelle laut Dokumentation:

```
STATUS LNPUBLIC NSFGetServerLatency(char far *ServerName,
DWORD  Timeout, DWORD far *retClientToServerMS,
DWORD far *retServerToClientMS, WORD far *ServerVersion);
```

- „übersetzt“ nach LotusScript:

```
Declare Private Function W32_NSFGetServerLatency _
    Lib "nnotes.dll" Alias "NSFGetServerLatency" _
    (ByVal serverName As LMBCS String, ByVal timeout As Long, _
    msToServer As Long, msFromServer As Long, _
    serverVersion As Integer) As Integer
```

assono

23 C-API – GetCAPIErrorMsg

- **GetCAPIErrorMsg**

```
Public Function GetCAPIErrorMsg(errorCode As Integer) As String
    buffer = String$(256, 0)
    length = OSLoadString(NULLHANDLE, errorCode, buffer, Len(buffer))
    If length > 0 Then
        GetCAPIErrorMsg = Left$(buffer, InStr(1, buffer, Chr$(0)) - 1)
    Else
        GetCAPIErrorMsg = "Unbekannter Fehler„
    End If
End Function
```

assono

23 C-API – GetNotesTempDirectory

- **GetNotesTempDirectory**

```
Public Function GetNotesTempDirectory() As String

    Dim notesTempDirectory As String * MAXPATH
    Dim pathLength As Integer

    pathLength = OSGetSystemTempDirectory(notesTempDirectory)

    GetNotesTempDirectory = Left$(notesTempDirectory, pathLength)
End Function
```

assono

23 C-API – OSGetSystemTempDirectory

- **OSGetSystemTempDirectory**

```
Public Function OSGetSystemTempDirectory(pathName As String) As Long

    Select Case session.Platform
        Case PLATFORM_W32
            OSGetSystemTempDirectory =_
                W32_OSGetSystemTempDirectory(pathName, MAXPATH - 1)
[…]
    End Select

End Function
```

assono

23 C-API – W32_OSGetSystemTempDirectory

- **W32_OSGetSystemTempDirectory**
- Schnittstelle laut Dokumentation:

```
???
```

Funktion ist undokumentiert, siehe:
http://www-10.lotus.com/ldd/bpmpblog.nsf/dx/tempfilestream-class

```
Declare Private Function W32_OSGetSystemTempDirectory _
    Lib "nnotes.dll" Alias "OSGetSystemTempDirectory"_
    (ByVal pathName As String, ByVal bufferLen As Integer)_
    As Integer
```

24 LS2J

- aus LotusScript direkt auf Java-Objekte zugreifen, Attribute lesen und ändern, Methoden aufrufen…
- Java kann einiges, was LotusScript nicht kann, z. B.
  - Netzwerkzugriff
  - Grafikverarbeitung
  - parallele Threads
- sehr viele fertige Bibliotheken, häufig Open Source
- im Gegensatz zur Windows-API plattformunabhängig

assono

24 LS2J – Reguläre Ausdrücke

- Beispiel: Reguläre Ausdrücke
- LotusScript: Like
- Formelsprache: @Matches und @Like
- relativ „ausdrucksschwach“
- Reguläre Ausdrücke in Java können viel, viel mehr…
- für einfache Fälle reicht java.lang.String.matches
- sonst gibt es auch java.util.regex.Pattern und java.util.regex.Matcher
- LS2J nutzen: `UseLSX "*javacon"`

assono

24 LS2J – Reguläre Ausdrücke in Java

Summary of regular-expression constructs

Construct	Matches
Characters	
x	The character *x*
\\	The backslash character
\0*n*	The character with octal value 0*n* (0 <= *n* <= 7)
\0*nn*	The character with octal value 0*nn* (0 <= *n* <= 7)
\0*mnn*	The character with octal value 0*mnn* (0 <= *m* <= 3, 0 <= *n* <= 7)
\x*hh*	The character with hexadecimal value 0x*hh*
\u*hhhh*	The character with hexadecimal value 0x*hhhh*
\x{*h...h*}	The character with hexadecimal value 0x*h...h* (Character.MIN_CODE_POINT <= 0x*h...h* <= Character.MAX_CODE_POINT)
\t	The tab character ('\u0009')
\n	The newline (line feed) character ('\u000A')
\r	The carriage-return character ('\u000D')
\f	The form-feed character ('\u000C')
\a	The alert (bell) character ('\u0007')
\e	The escape character ('\u001B')
\cx	The control character corresponding to *x*

http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

assono

24 LS2J – Reguläre Ausdrücke in Java

Character classes

`[abc]`	a, b, or c (simple class)
`[^abc]`	Any character except a, b, or c (negation)
`[a-zA-Z]`	a through z or A through Z, inclusive (range)
`[a-d[m-p]]`	a through d, or m through p: `[a-dm-p]` (union)
`[a-z&&[def]]`	d, e, or f (intersection)
`[a-z&&[^bc]]`	a through z, except for b and c: `[ad-z]` (subtraction)
`[a-z&&[^m-p]]`	a through z, and not m through p: `[a-lq-z]`(subtraction)

Predefined character classes

`.`	Any character (may or may not match line terminators)
`\d`	A digit: `[0-9]`
`\D`	A non-digit: `[^0-9]`
`\h`	A horizontal whitespace character: `[ \t\xA0\u1680\u180e\u2000-\u200a \u202f\u205f\u3000]`
`\H`	A non-horizontal whitespace character: `[^\h]`
`\s`	A whitespace character: `[ \t\n\x0B\f\r]`
`\S`	A non-whitespace character: `[^\s]`
`\v`	A vertical whitespace character: `[\n\x0B\f\r\x85\u2028\u2029]`
`\V`	A non-vertical whitespace character: `[^\v]`
`\w`	A word character: `[a-zA-Z_0-9]`
`\W`	A non-word character: `[^\w]`

http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

assono

24 LS2J – Reguläre Ausdrücke in Java

Boundary matchers

^	The beginning of a line
$	The end of a line
\b	A word boundary
\B	A non-word boundary
\A	The beginning of the input
\G	The end of the previous match
\Z	The end of the input but for the final terminator, if any
\z	The end of the input

Linebreak matcher

\R	Any Unicode linebreak sequence, is equivalent to `\u000D\u000A|[\u000A\u000B\u000C\u000D\u0085\u2028\u2029]`

http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

assono

24 LS2J – Reguläre Ausdrücke in Java

Greedy quantifiers

X?	*X*, once or not at all
*X**	*X*, zero or more times
X+	*X*, one or more times
X{*n*}	*X*, exactly *n* times
X{*n*,}	*X*, at least *n* times
X{*n*,*m*}	*X*, at least *n* but not more than *m* times

Reluctant quantifiers

X??	*X*, once or not at all
*X**?	*X*, zero or more times
X+?	*X*, one or more times
X{*n*}?	*X*, exactly *n* times
X{*n*,}?	*X*, at least *n* times
X{*n*,*m*}?	*X*, at least *n* but not more than *m* times

Possessive quantifiers

X?+	*X*, once or not at all
*X**+	*X*, zero or more times
X++	*X*, one or more times
X{*n*}+	*X*, exactly *n* times
X{*n*,}+	*X*, at least *n* times
X{*n*,*m*}+	*X*, at least *n* but not more than *m* times

http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

assono

24 LS2J – Reguläre Ausdrücke in Java

Logical operators	
XY	*X* followed by *Y*
X|*Y*	Either *X* or *Y*
(*X*)	X, as a capturing group
Back references	
n	Whatever the *n*th capturing group matched
k<*name*>	Whatever the named-capturing group "name" matched
Quotation	
\	Nothing, but quotes the following character
\Q	Nothing, but quotes all characters until \E
\E	Nothing, but ends quoting started by \Q

http://docs.oracle.com/javase/8/docs/api/java/util/regex/Pattern.html

24 LS2J – Reguläre Ausdrücke

- String.matches benutzen:

```
Function MatchesJavaRegExp(ByVal testString As String,_
    ByVal pattern As String) As Boolean

Dim javaSession As JavaSession
Dim stringClass As JavaClass
Dim passwordString As JavaObject

Set javaSession = New JavaSession()
Set stringClass = javaSession.GetClass("java/lang/String")
Set passwordString =
stringClass.CreateObject("(Ljava/lang/String;)V", testString)
MatchesJavaRegExp = passwordString.matches(pattern)
```

24 LS2J – Reguläre Ausdrücke

- Fehlerbehandlung

```
Dim javaError As JavaError
Dim javaErrorMessage As String

On Error Goto errorHandler

[…]

errorHandler:
    If Not javaSession Is Nothing Then
        Set javaError = javaSession.GetLastJavaError()
        If javaError.errorMsg <> "" Then
            javaErrorMessage = javaError.ErrorMsg & CRLF & CRLF &_
                javaError.StackTrace
        End If
        javaSession.ClearJavaError
    End If
```

assono

24 LS2J – Reguläre Ausdrücke

- mit Pattern und Matcher-Objekten:

```
Function MatchesJavaRegExpAdvanced(ByVal testString As String,_
ByVal pattern As String, ByVal flags As Integer) As Boolean

Dim javaSession As JavaSession
Dim patternClass As JavaClass
Dim patternObject As JavaObject
Dim matcherObject As JavaObject

Set javaSession = New JavaSession()
Set patternClass = javaSession.GetClass("java/util/regex/Pattern")
Set patternObject = patternClass.compile(pattern, flags)
Set matcherObject = patternObject.matcher(testString)
MatchesJavaRegExpAdvanced = matcherObject.matches()
```

24 LS2J – Reguläre Ausdrücke

- Flags für compile() (Dokumentation):

```
Const MF_UNIX_LINES = 1' enables Unix lines mode.
Const MF_CASE_INSENSITIVE = 2' enables case-insensitive matching.
Const MF_COMMENTS = 4' permits whitespace and comments in pattern.
Const MF_MULTILINE = 8' enables multiline mode.
Const MF_LITERAL = 16' enables literal parsing of the pattern.
Const MF_DOTALL = 32' enables dotall mode.
Const MF_UNICODE_CASE = 64' enables Unicode-aware case folding.
Const MF_CANON_EQ = 128' enables canonical equivalence.
```

24 LS2J – Reguläre Ausdrücke

- Verwendung:

```
allStats = session.SendConsoleCommand(currentDB.Server, "sh stat")
allStats = FullTrim(Split(allStats, Chr$(13)))

pattern = "Platform.*„
Do
    pattern = InputBox$("Welche Statistiken ausgeben?", "", pattern)
    If pattern = "" Then Exit Sub

    ForAll stat In allStats
        If MatchesJavaRegExp(stat, pattern) Then
            Print stat
        End If
    End ForAll
Loop
```

assono

25 NotesNoteCollection – Ansicht aktueller Dokumente

- alle Dokumente aus den letzten 7 Tagen
- mehrere Alternativen
  - @Now/@Today in der Selektionsformel
  - täglich laufender Agent ändert Feld in den Dokumenten
  - täglich laufender Agent schiebt sie in einen Ordner
  - täglich laufender Agent ändert die Selektionsformel

25 NotesNoteCollection – Selektionsformel ändern

- Beispiel Selektionformel:

```
_currentDate := @Today;
_days := 5;

_delta := (@Date(WiedervorlageAm) - _currentDate) / 86400;
SELECT Form = "Wiedervorlage" & (@Abs(_delta) < _days)
```

- daraus macht der Agent am 13. April 2016:

```
_currentDate := [13.04.2016];
_days := 5;

_delta := (@Date(WiedervorlageAm) - _currentDate) / 86400;
SELECT Form = "Wiedervorlage" & (@Abs(_delta) < _days)
```

- Vorteil dieser Form: auch wenn die Ansicht aus der Schablone aktualisiert wird, werden die richtigen Dokumente angezeigt.

25 NotesNoteCollection – Selektionsformel ändern

- Beispiel – Alle Ansichten durchlaufen

```
Set viewsColl = currentDB.CreateNoteCollection(False)
viewsColl.SelectViews = True
Call viewsColl.BuildCollection

viewNoteID = viewsColl.GetFirstNoteID
For i = 1 To viewsColl.Count
    Set viewDoc = currentDB.GetDocumentByID(viewNoteID)
    Set view = currentDB.GetView(StrLeft$(_
        viewDoc.GetItemValue("$TITLE")(0) & "|", "|"))
    If Not view Is Nothing Then
        Call UpdateSelectionFormulaForToday(view)
        viewNoteID = viewsColl.GetNextNoteID(viewNoteID)
    End If
Next
```

25 NotesNoteCollection – Selektionsformel ändern

- Beispiel – Selektionformel aktualisieren

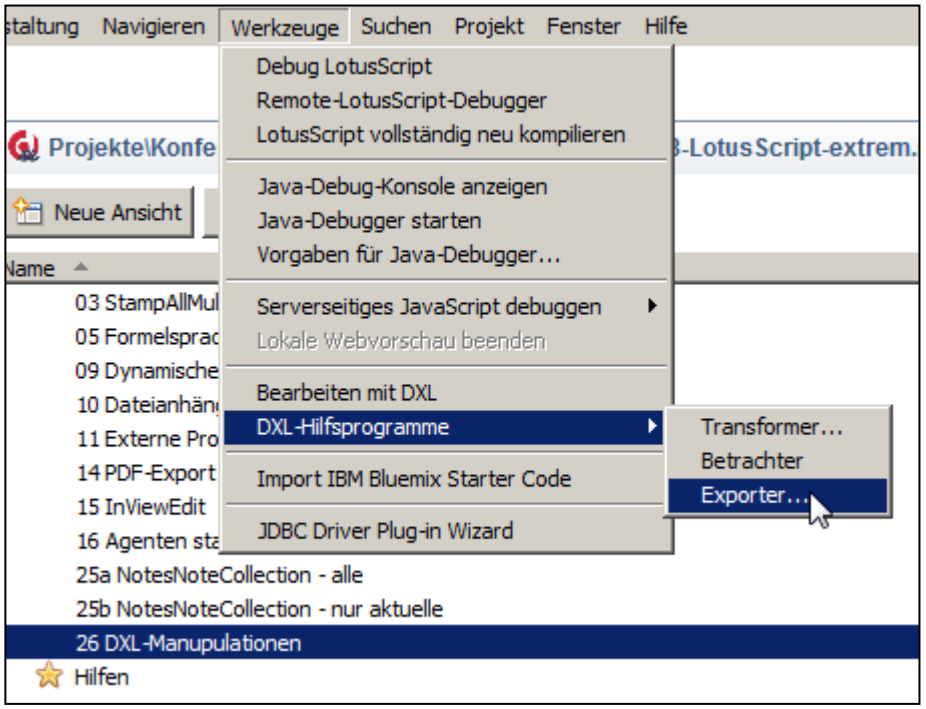
```
formula = view.SelectionFormula
posBegin = InStr(formula, "_currentDate")
If posBegin = 1 Then
    posEnd = InStr(posBegin, formula, ";")
    remainder = Mid$(formula, posEnd)
    newFormula = "_currentDate" & " := [" & CStr(Today) &_
        "]" &  remainder
    If view.SelectionFormula <> newFormula Then
        Print "geänderte Ansicht: '" & view.Name & "'"
        view.SelectionFormula = newFormula
        Call view.Refresh()
    End If
End If
```

26 DXL-Manipulationen

- Gestaltungselement wie Agent, Ansicht oder Ordner in eine andere Datenbank „injizieren“
- dabei anpassen, z. B. Namen oder „Zielkoordinaten“ wie eine Replik-ID ändern
- 1. Schritt: DXL-Export
- 2. Schritt: DXL anpassen
- 3. Schritt: Installationsagent schreiben
- 4. Schritt: DXL in Konstante packen
- 5. Schritt: ausführen und testen

26 DXL-Manipulationen – 1. Schritt: DXL-Export

- Im Domino Designer gewünschtes Element markieren und mit Werkzeuge – DXL-Hilfsprogramme – Exporter… exportieren



26 DXL-Manipulationen – 2. Schritt: DXL anpassen

- Im Texteditor DXL anpassen, z. B. <database>, <databaseinfo> und <launchsettings> entfernen, DOCTYPE anpassen und Platzhalter einbauen:

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE database SYSTEM 'xmlschemas/domino_9_0_1.dtd'>
<database xmlns='http://www.lotus.com/dxl' version='9.0' maintenanceversion='1.5'
 replicaid='C1257F86006FC2A6' path='Projekte\Konferenzen\EntwicklerCamp-2016\EC16-T288-LotusScript-extrem.nsf'
 title='EC16 T288 LotusScript extrem!' categories='assono,EntwicklerCamp,Konferenzen'
 allowstoredforms='false' multilingual='true' increasemaxfields='true' uselz1='true'>
<databaseinfo dbid='C1257F8C0020480A' odsversion='52' diskspace='4194304'
 percentused='90.606689453125' numberofdocuments='60'><datamodified><datetime
 dst='true'>20160410T154528,04+02</datetime></datamodified><designmodified
><datetime dst='true'>20160410T155409,80+02</datetime></designmodified></databaseinfo>
<launchsettings><noteslaunch whenopened='openframeset' frameset='Hauptrahmengruppe'/></launchsettings>
<view name='26 DXL-Manupulationen' showinmenu='false' noreplace='true' publicaccess='false'
 designerversion='8.5.3' unreadmarks='none' onopengoto='lastopened' onrefresh='displayindicator'
 headers='simple' opencollapsed='true' showresponsehierarchy='false' showmargin='true'
 shrinkrows='true' extendlastcolumn='true' showhierarchies='false' unreadcolor='black'
```

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE view SYSTEM 'xmlschemas/domino_8_5_3.dtd'>
<view name='<%VIEW_NAME%>' showinmenu='false' norepla
 designerversion='8.5.3' unreadmarks='none' onopengot
 headers='simple' opencollapsed='true' showresponsehi
```

assono

26 DXL-Manipulationen – 3. Schritt: Installationsagent

- alles vorbereiten, …

```
Set mailDB = New NotesDatabase("", "")
Call mailDB.OpenMail()
If Not mailDB.IsOpen() Then _
    Error 32002, "Mail-DB konnte nicht geöffnet werden."

viewName = Trim(Inputbox$("Unter welchem Namen soll Ansicht " &_
    "in Ihrer Mail-Datenbank installiert werden?",_
    "Installiere Ansicht in Mail-Datenbank",_
    "Kopiere in die Demo-DB"))
If viewName = "" Then Exit Sub

dxlString = Replace(VIEW_DXL, "<%VIEW_NAME%>", viewName)
dxlString = Replace(dxlString, "<%DEMO_DB_REPLICA_ID%>",_
    currentDB.ReplicaID)
```

assono

26 DXL-Manipulationen – 3. Schritt: Installationsagent

- … DXL importieren …

```
Set importer = session.CreateDXLImporter()
importer.DesignImportOption = DXLIMPORTOPTION_REPLACE_ELSE_CREATE
importer.ReplicaRequiredForReplaceOrUpdate = False
importer.InputValidationOption = VALIDATE_ALWAYS
importer.CompileLotusScript = True
Call importer.Import(dxlString, mailDB)

Messagebox importer.Log, 0,  importer.ImportedNoteCount &_
    " Nodes importiert"
```

assono

26 DXL-Manipulationen – 3. Schritt: Installationsagent

- … und Gestaltungselement signieren

```
If importer.ImportedNoteCount > 0 Then
    noteID = importer.GetFirstImportedNoteId
    Set viewNote = mailDB.GetDocumentByID(noteID)
    Call viewNote.Sign()
    Call viewNote.Save(True, False)
End If
```

26 DXL-Manipulationen – 4. Schritt: DXL-Konstante

- DXL in eine Konstante packen (oder in ein Dokument oder …)

```
Private Const VIEW_DXL = {<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE view SYSTEM 'xmlschemas/domino_8_5_3.dtd'>
<view name='<%VIEW_NAME%>' showinmenu='false' noreplace='true'
publicaccess='false'  designerversion='8.5.3' unreadmarks='none'
onopengoto='lastopened' onrefresh='displayindicator'

[…]

</formula></code></column></view>}
```

assono

26 DXL-Manipulationen – 5. Schritt: ausführen & testen

- mehrfach ausführen und gründlich mit verschiedenen Benutzern (unterschiedliche Berechtigungen) testen
- Benutzer braucht Designer-Rechte oder Agent wird auf dem Server mit höheren Berechtigungen ausgeführt
- **Bonus:** Verfahren funktioniert auch, wenn die Ziel-Datenbank eine versteckte Gestaltung hat

27 RichText mit Tabellen

- RichText-Item erstellen und Absatzstile vorbereiten

```
Set rtHelper = CreateRichTextHelperWithNewItem(doc, itemName)
Call rtHelper.SetSize(9)

Dim pstyles(1 To 5) As NotesRichTextParagraphStyle
For i = 1 To 5
    Set pstyles(i%) = session.CreateRichTextParagraphStyle
    If i = 1 Then ' first column
        pstyles(i%).Alignment = ALIGN_LEFT
        pstyles(i%).RightMargin = RULER_ONE_INCH * 2.2
    Else
        pstyles(i%).Alignment = ALIGN_RIGHT
        pstyles(i%).RightMargin = RULER_ONE_INCH * 0.75
    End If
    pstyles(i%).LeftMargin = 0
    pstyles(i%).FirstLineLeftMargin = 0
Next
```

assono

27 RichText mit Tabellen

- Tabelle erstellen und füllen

```
Set bodyRTItem = rtHelper.GetRTItem()
Call bodyRTItem.AppendTable(UBound(servers) - LBound(servers) +_
    2, 5, , , pstyles)

Set rtnav = bodyRTItem.CreateNavigator
Call rtnav.FindFirstElement(RTELEM_TYPE_TABLECELL)

Call rtHelper.WriteToCellAndAdvance("Server", rtnav)
Call rtHelper.WriteToCellAndAdvance("1. Test", rtnav)
Call rtHelper.WriteToCellAndAdvance("2. Test", rtnav)
Call rtHelper.WriteToCellAndAdvance("3. Test", rtnav)
Call rtHelper.WriteToCellAndAdvance("Mittelwert", rtnav)

For i = LBound(servers) To UBound(servers)
    serverLatencies = allServerLatencies(servers(i))
    Call WriteToRichTextTable(rtHelper, bodyRTItem, rtNav,_
        servers(i), serverLatencies)
Next
```

27 RichText mit Tabellen

- **WriteToRichTextTable**

```
Call body.BeginInsert(rtnav)
Call rtHelper.AppendText(server)
Call body.EndInsert
Call rtnav.FindNextElement(RTELEM_TYPE_TABLECELL)

If serverLatencies(0) > 80 Then
    Call rtHelper.SetColor(COLOR_RED)
ElseIf serverLatencies(0) > 40 Then
    Call rtHelper.SetColor("ff8000")
Else
    Call rtHelper.SetColor(COLOR_DARK_GREEN)
End If

Call rtHelper.WriteToCellAndAdvance(data(1), rtnav)
[…]

Call rtHelper.SetColor(COLOR_BLACK)
```

assono

27 RichText mit Tabellen

- **WriteToCellAndAdvance**

```
Call rtItem.BeginInsert(rtnav)
Call rtItem.AppendText(textToAppend)
Call rtItem.EndInsert
Call rtnav.FindNextElement(RTELEM_TYPE_TABLECELL)
```

assono

27 RichText mit Tabellen

- Bericht erstellen und Dokument neu öffnen

```
Call WriteReportToRichTextItem(servers, allServerLatencies,_
currentDoc, "Bericht")
Call currentDoc.ReplaceItemValue("Testzeitpunkt", Now)

' reopen document in order to update RichText field
Call currentDoc.ReplaceItemValue("SaveOptions", "0")
Call currentDoc.ComputeWithForm(False, False)
Call uidoc.Close(True)
Call uiws.EditDocument(True, currentDoc)
Call currentDoc.RemoveItem("SaveOptions")
```

28 Replikation

- Replikationshistorie löschen und Replizieren

```
Print "Lösche Replikationshistorie auf " & currentDB.Server
Set replicationInfo = currentDB.ReplicationInfo
Call replicationInfo.ClearHistory()

ForAll server In servers
    Set targetDB = New NotesDatabase("", "")
    If targetDB.OpenByReplicaID(server, replicaID) Then
        Print "Lösche Replikationshistorie auf " & server
        Set replicationInfo = targetDB.ReplicationInfo
        Call replicationInfo.ClearHistory()
    End If

    Print "Repliziere mit Server " & server
    Call currentDB.Replicate(CStr(server))
End ForAll
```

29 NotesDOMParser

- NotesNoteCollection aller Gliederungen erstellen, DOM-Parser erstellen und laufen lassen

```
Set outlinesColl = currentDB.CreateNoteCollection(False)
outlinesColl.SelectOutlines = True
Call outlinesColl.BuildCollection

Set domParser = session.CreateDOMParser()

Set dxlExporter = session.CreateDXLExporter
Call dxlExporter.SetInput(outlinesColl)
Call dxlExporter.SetOutput(domParser)
Call dxlExporter.Process
```

assono

29 NotesDOMParser

- Ergebnis des Parsers ist NotesDOMDocumentNode
- alle <outline>-Elemente durchlaufen

```
Set outlinesList = domParser.Document.GetElementsByTagName("outline")
For i = 1 To outlinesList.NumberOfEntries
    Set outlineNode = outlinesList.GetItem(i)
    outlineName = GetAttributeValue(outlineNode, "name")

    Set outlineChildNode = outlineNode.FirstChild
    Do While Not outlineChildNode.IsNull
        Call ExamineOutlineChildNode(outlineChildNode,_
            currentReplicaID, outlineName)
        Set outlineChildNode = outlineChildNode.NextSibling
    Loop
Next
```

assono

29 NotesDOMParser

- Im<outline>-Element alle Kinder durchlaufen

```
If node.NodeName = "outlineentry" Then
    outlineEntryLabel = GetAttributeValue(node, "label")

    Set outlineEntryChildNode = node.FirstChild
    Do While Not outlineEntryChildNode.IsNull
        Call ExamineOutlineEntryChild(outlineEntryChildNode,_
            currentReplicaID, outlineEntryLabel, outlineName)

        Set outlineEntryChildNode = outlineEntryChildNode.NextSibling
    Loop
End If
```

assono

29 NotesDOMParser

- Im<imageref>-Elemente untersuchen

```
If node.NodeName = "imageref" Then
    imagerefDBReplicaID = GetAttributeValue(node, "database")
    If imagerefDBReplicaID <> currentReplicaID Then
        Print "Falsche Replik-ID (" & imagerefDBReplicaID &_
            ") im <imageref>-Elements des Gliederungseintrags '" &_
            outlineEntryLabel & "' der Gliederung '" & outlineName &_
            "'"
    End If
End If
```

assono

29 NotesDOMParser

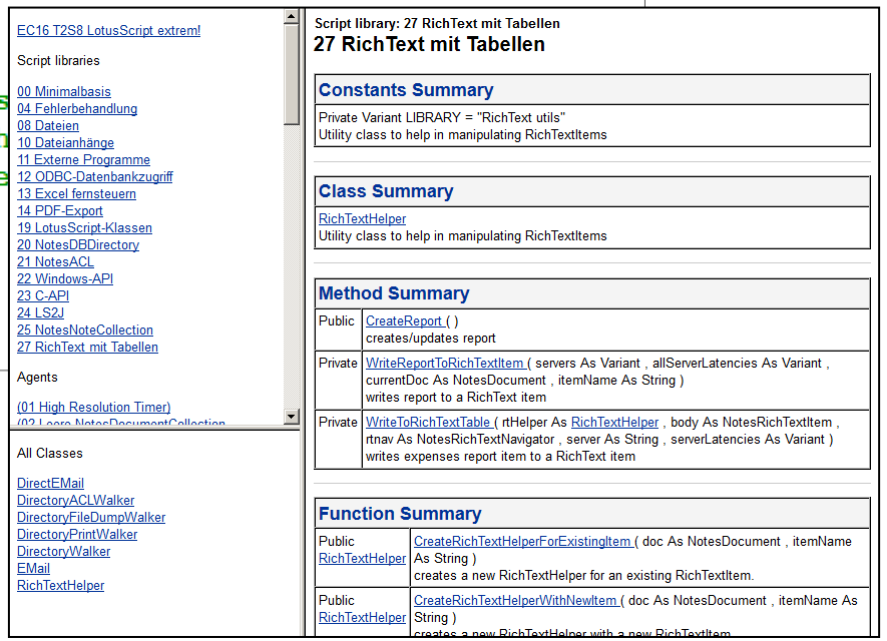
- ein Attribut in einer DOM-Node suchen

```
GetAttributeValue = ""
Set attributes = node.Attributes
For i = 1 To attributes.NumberOfEntries
    Set attributeNode = attributes.GetItem(i)
    If attributeNode.NodeName = attributeName Then
        GetAttributeValue = attributeNode.NodeValue
    Exit Function
    End If
Next
```

30 Werkzeuge – LotusScript.doc

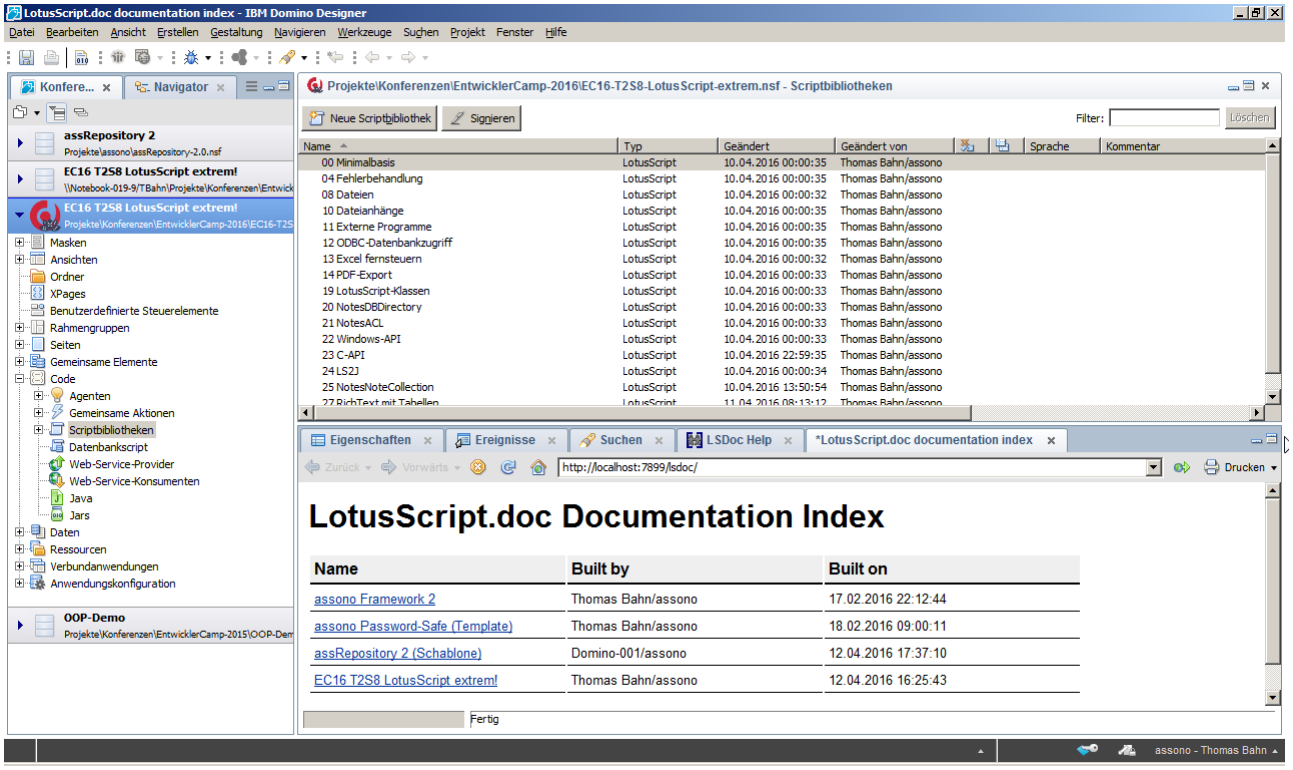
- LotusScript.doc 2.0.0 (ls.doc) – http://lekkimworld.com/2009/12/11/im_very_happy_to_announce_that_lotusscript_doc_v_2_is_finally_done.html
- LotusScript-Dokumentation aus dem Code generieren

```
Private Sub WriteReportToRichTextItem(servers As Variant, allServerLatencies List As
    '/**
    ' * writes report to a RichText item
    ' *
    ' * @param       servers array of server names
    ' * @param       allServerLatencies list of arrays
    ' * @param       currentDoc document containing th
    ' * @param       itemName name of the RichText ite
    ' *
    ' * @author      Thomas Bahn <tbahn@assono.de>
    ' * @version     2016-04-10
    ' */
```



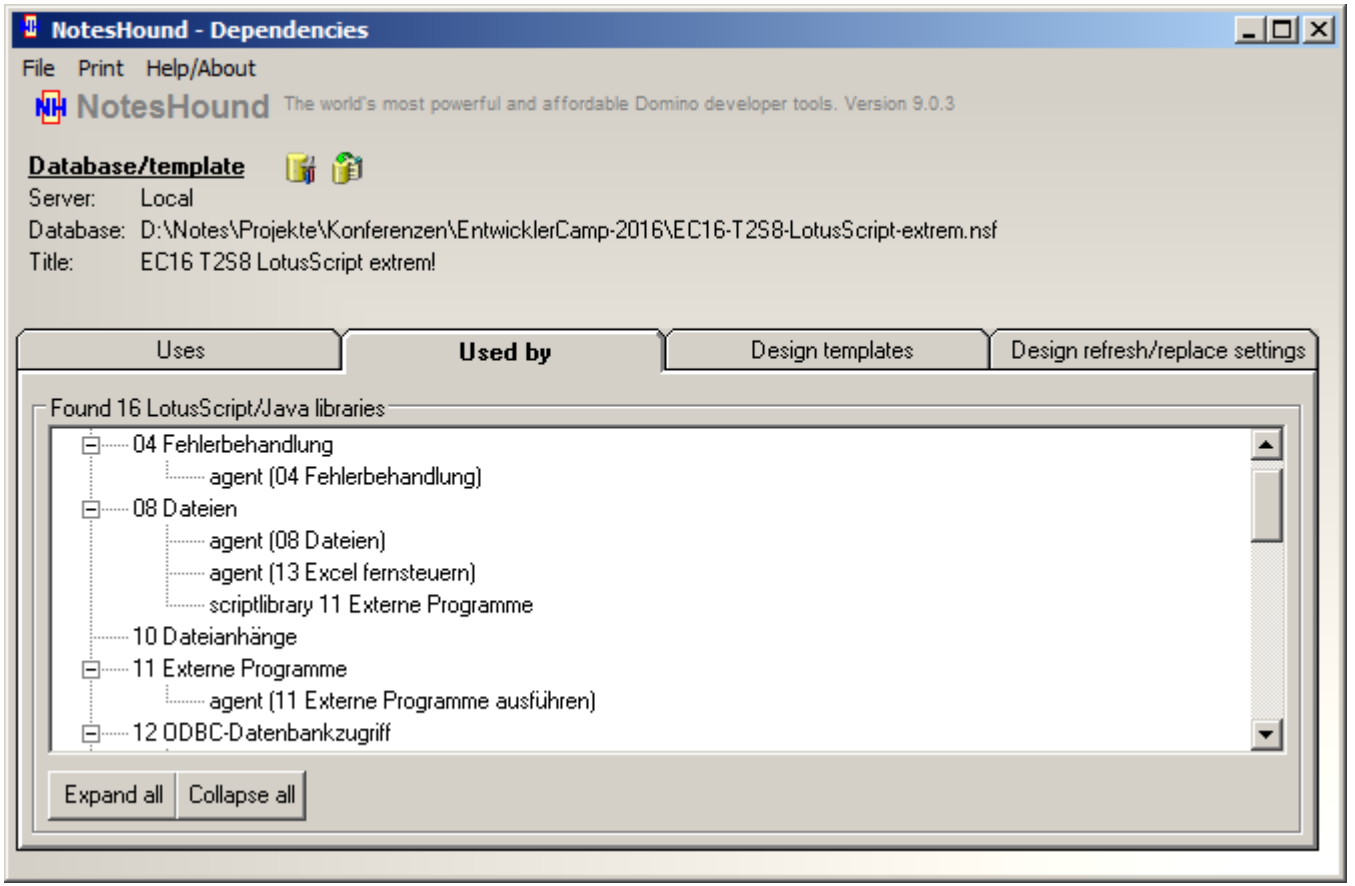
30 Werkzeuge – LotusScript.doc

- Add LSDoc Support to Domino Designer on Eclipse with Custom Plug-ins – blog.mindoo.com/web/blog.nsf/dx/22.07.2010223240KLES4H.htm



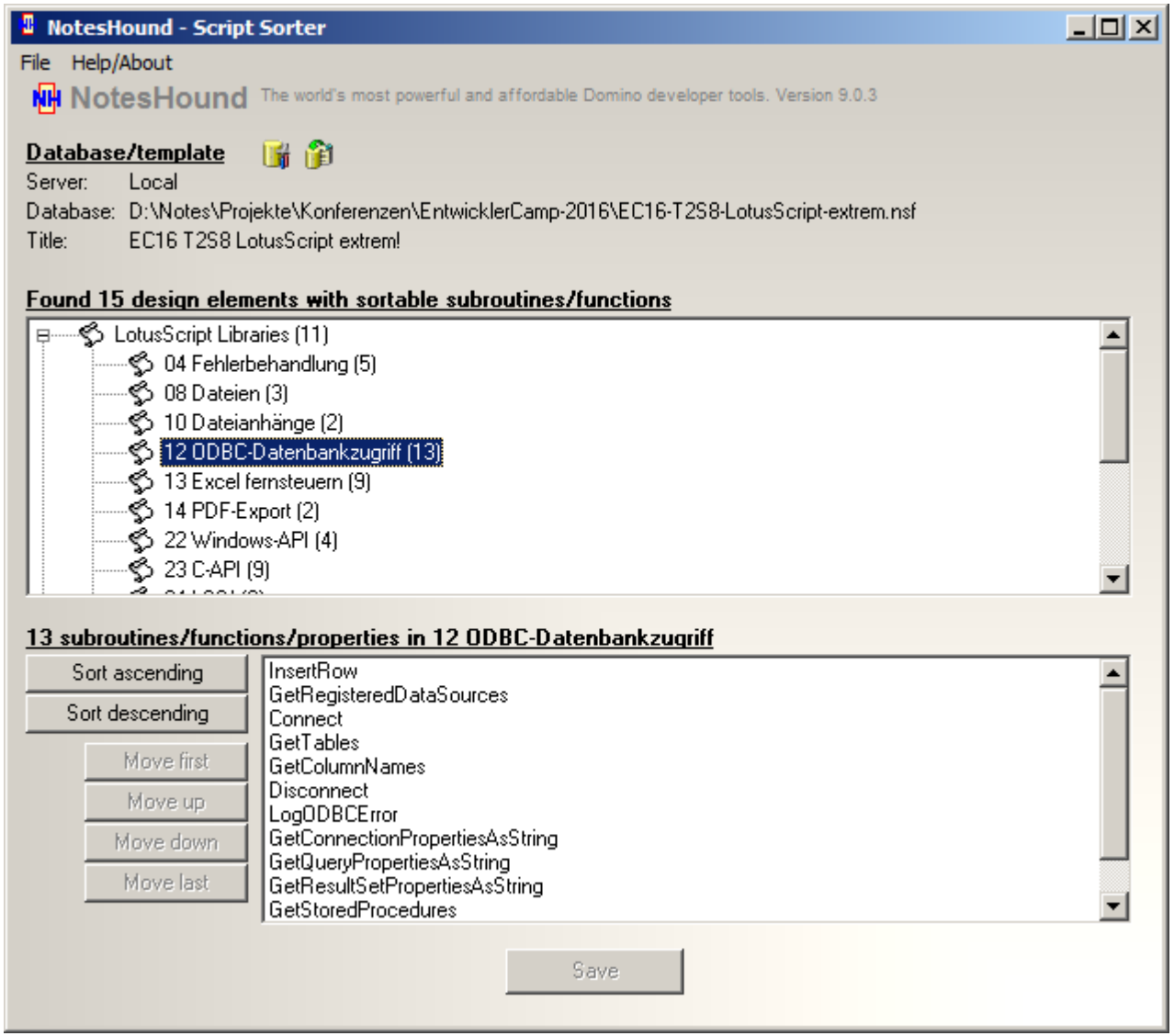
30 Werkzeuge – NotesHound

- NotesHound – http://noteshound.com
- 18 Werkzeuge, darunter Dependencies



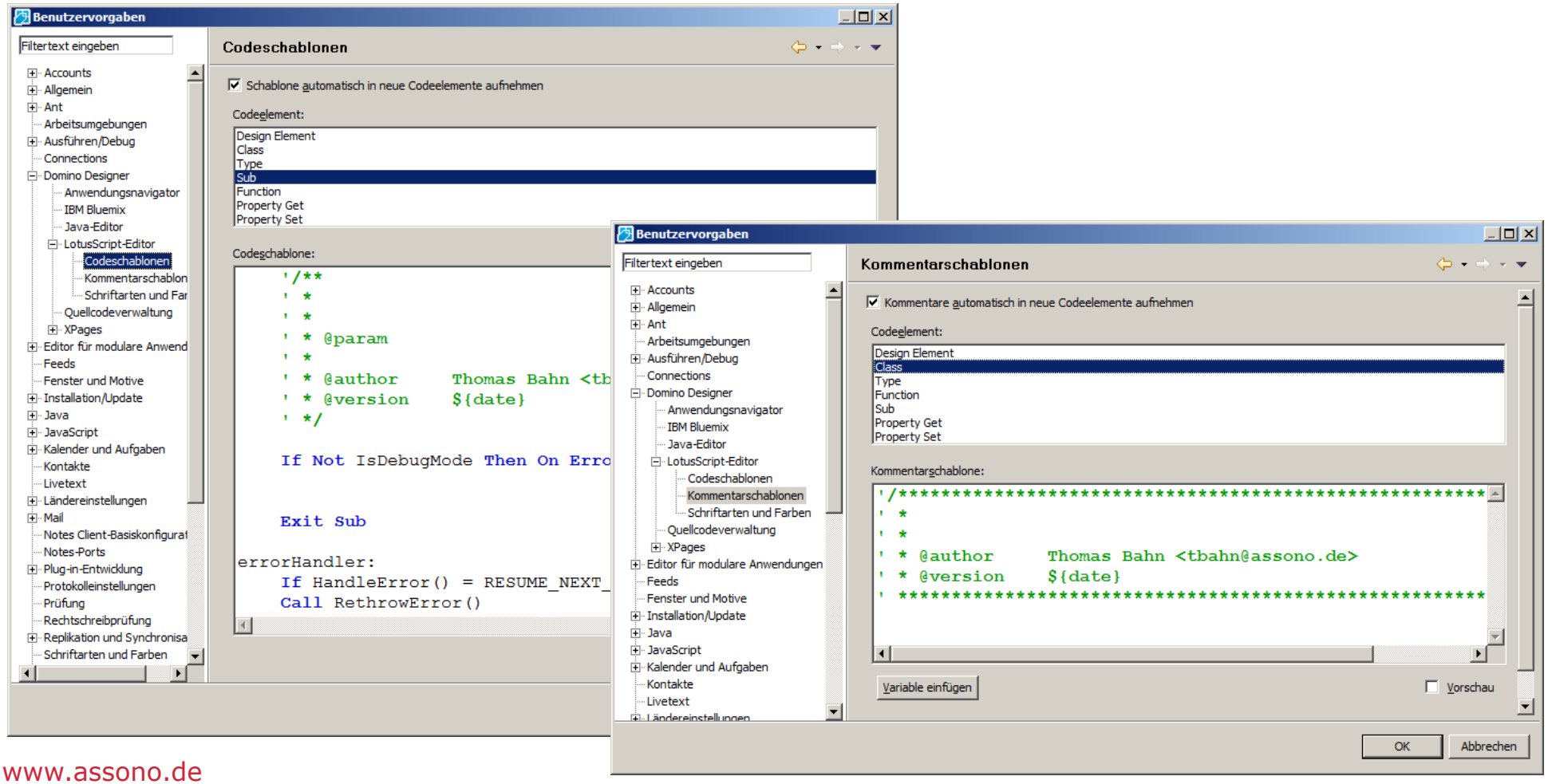
30 Werkzeuge – NotesHound

- Script Sorter



30 Werkzeuge – Domino Designer

- Domino Designer
- Vorlagen für Code-Elemente und Kommentare



Fragen?

- jetzt stellen – oder später:

tbahn@assono.de
www.assono.de/blog
04307/900-401

assono
IT-Consulting & Solutions

- Folien unter:
  www.assono.de/blog/d6plinks/EC-2016-LotusScript-**extrem**