Richtextmanipulation in LotusScript

EC15 | Track 1 – Session 6 | Torsten Link (Tode)

Richtextmanipulation in LotusScript

- Einführung: Die NotesRichtextItem- Klasse und ihre Kinder
  - NotesRichtextItem
  - NotesRichtextStyle, NotesRichtextParagraphStyle
  - NotesRichtextNavigator
- Manipulation von Richtext mit den Standardmitteln
  - Hinzufügen von Text
  - Formatieren mit RichtextStyle
  - Suchen und Ersetzen mit FindAndReplace
  - Hinzufügen von Tabellen
- Weiterführende Manipulation mittels „Vorlagemasken“ oder „Vorlagedokumenten“
- Herausforderung: Änderungen im Frontend sichtbar machen
- HTML für formatierte Mails verwenden
- Hotspots mit „dynamischem Code“ versenden: DXL- Manipulation

Richtextmanipulation in LotusScript

Die NotesRichtextItem- Klasse

- Erbt von NotesItem, damit auch alle Eigenschaften / Methoden
- Alle Methoden mit dem Hinweis (from NotesItem) nur lesend verwenden, Schreibzugriffe behandeln das Item wie ein NotesItem – alles, was ein RichtextItem ausmacht geht verloren
- Einzige direkt Für RichtextItem relevante Eigenschaft:
- EmbeddedObjects (Out of Scope für diese Session)
- Kann durch NotesDocument.GetFirstitem oder New NotesRichtextItem initialisiert werden.
- Wichtige Methode: Compact() um das erstellen zu vieler RTItems zu verhindern!

ACHTUNG: NotesRichtextItems sind sehr speziell. Ein Beispiel hierfür folgt sofort

Properties

DateTimeValue (from NotesItem)
EmbeddedObjects
IsAuthors (from NotesItem)
IsEncrypted (from NotesItem)
IsNames (from NotesItem)
IsProtected (from NotesItem)
IsReaders (from NotesItem)
IsSigned (from NotesItem)
IsSummary (from NotesItem)
LastModified (from NotesItem)
Name (from NotesItem)
Parent (from NotesItem)
SaveToDisk (from NotesItem)
Text (from NotesItem)
Type (from NotesItem)
ValueLength (from NotesItem)
Values (from NotesItem)

Richtextmanipulation in LotusScript

Die NotesRichtextItem- Klasse

```
On Error GoTo ErrorRoutine
'================================================
Dim ws As New NotesUIWorkspace

Dim doc As NotesDocument
Dim rtItem As NotesRichTextItem

Set doc = New NotesDocument( g_dbCurrent )
Call doc.Replaceitemvalue( "Form", "defaultForm" )
Set rtItem = New NotesRichTextItem( doc, "Body" )

Call rtItem.Appendtext( "Das ist mein erster Richtext" )
Call rtItem.Addnewline( 2 )
Call rtItem.Appendtext( "Sogar zweizeilig..." )

Call ws.Editdocument( True, doc )

'================================================
EndOfRoutine:
Exit Sub
ErrorRoutine:
If ErrorHandler (GetThreadInfo (LSI_THREAD_PROC), GetThreadInfo (LSI_THREAD_CALLPROC)) = ERR_TOP_OF_STACK Then
    Resume EndOfRoutine
End If
```

Richtextmanipulation in LotusScript

Die NotesRichtextItem- Klasse

```
On Error GoTo ErrorRoutine
'=================================================
Dim ws As New NotesUIWorkspace

Dim doc As NotesDocument
Dim rtItem As NotesRichTextItem

Set doc = New NotesDocument( g_dbCurrent )
Call doc.Replaceitemvalue( "Form", "defaultForm" )
Set rtItem = New NotesRichTextItem( doc, "Body" )

Call rtItem.Appendtext( "Das ist mein erster Richtext" )
Call rtItem.Addnewline( 2 )
Call rtItem.Appendtext( "Sogar zweizeilig..." )

Call rtItem.Update()

Call ws.Editdocument( True, doc )

'=================================================
EndOfRoutine:
Exit Sub
ErrorRoutine:
If ErrorHandler (GetThreadInfo (LSI_THREAD_PROC), GetThreadInfo (LSI_THREAD_CALLPROC)) = ERR_TOP_OF_STACK Then
    Resume EndOfRoutine
End If
```

ALSO: Immer Update verwenden, um Änderungen tatsächlich anzuwenden.

Richtextmanipulation in LotusScript

Die NotesRichtextStyle- Klassen

- Kann verwendet werden, um -in ziemlich eingeschränktem Umfang- die Formatierung von Text in einem RichtextItem zu beeinflussen.
- Wird über NotesSession.Createrichtextstyle generiert oder per Style- Property aus einer NotesRichtextRange ausgelesen (dazu später mehr)
- NotesColor erlaubt einen aus 240 verschiedenen Werten, 16 davon sind als LotusScript- Konstanten namentlich ansprechbar. ACHTUNG: Trotz Methode „setRGB“ der Klasse NotesColor KEINE freie Farbwahl
- Mit NotesFont verhält es sich ähnlich, siehe Hilfe…

Properties

Bold

Effects

FontSize

IsDefault

Italic

NotesColor

NotesFont

Parent

PassThruHTML

Strikethrough

Underline

ACHTUNG: Styles werden (in älteren Versionen) nicht immer zugewiesen, wenn man es vom Code her erwarten würde

Richtextmanipulation in LotusScript

Die NotesRichtextStyle- Klasse

```
On Error GoTo ErrorRoutine
'=================================================
Dim ws As New NotesUIWorkspace
Dim doc As NotesDocument
Dim rtItem As NotesRichTextItem
Dim rtStyle As NotesRichTextStyle

Set doc = New NotesDocument( g_dbCurrent )
Call doc.Replaceitemvalue( "Form", "defaultForm" )
Set rtItem = New NotesRichTextItem( doc, "Body" )

Set rtStyle = g_ses.Createrichtextstyle()
rtStyle.Bold = True
rtStyle.Notescolor = Color_red

Call rtItem.Appendstyle( rtStyle )
Call rtItem.Appendtext( "Das ist mein erster bunter Richtext" )

Call rtItem.Update()

Call ws.Editdocument( True, doc )

'=================================================
EndOfRoutine:
Exit Sub
ErrorRoutine:
If ErrorHandler (GetThreadInfo (LSI_THREAD_PROC), GetThreadInfo (LSI_THREAD_CALLPROC)) = ERR_TOP_OF_STACK Then
    Resume EndOfRoutine
End If
```

```
'- Version 8.5.x und älter
Call rtItem.Addnewline( 1 )
'- oder
Call rtItem.Appendtext( " " )
```

An ein leeres Item konnte man keinen Style anhängen

Richtextmanipulation in LotusScript

Die NotesRichtextParagraphStyle- Klasse

- Formatierungen für Paragraphen:
- Ausrichtung
- Ränder
- Zeilenabstände
- Tabs

Properties

Alignment

FirstLineLeftMargin

InterLineSpacing

LeftMargin

Pagination

RightMargin

SpacingAbove

SpacingBelow

Tabs

Methods

ClearAllTabs

SetTab

SetTabs

Richtextmanipulation in LotusScript

Die NotesRichtextNavigator- Klasse

- Wird verwendet, um durch ein RichtextItem zu navigieren.
- Findet Elemente der Typen
  - RTELEM_TYPE_DOCLINK (5)
  - RTELEM_TYPE_FILEATTACHMENT (8)
  - RTELEM_TYPE_OLE (9)
  - RTELEM_TYPE_SECTION (6)
  - RTELEM_TYPE_TABLE (1)
  - RTELEM_TYPE_TABLECELL (7)
  - RTELEM_TYPE_TEXTPARAGRAPH (4)
  - RTELEM_TYPE_TEXTRUN (3)
- Die Methoden setzen intern einen virtuellen „Cursor“ auf den Beginn des gefundenen (Range)- Objektes
- Mit nachfolgendem NotesRichtextItem.CreateRange kann das Range- Objekt ausgelesen werden
- Methoden geben True oder False zurück, je nachdem ob das gewünschte Element gefunden wurde oder nicht

Methods

Clone
FindFirstElement
FindFirstString
FindLastElement
FindNextElement
FindNextString
FindNthElement
GetElement
GetFirstElement
GetLastElement
GetNextElement
GetNthElement
SetCharOffset
SetPosition
SetPositionAtEnd

Richtextmanipulation in LotusScript

Die NotesRichtextRange- Klasse

- Die „Arbeiterklasse“
- Gebildet über NotesRichtextItem.CreateRange
- „interagiert“ mit der NotesRichtextNavigator- Klasse
  reagiert auf Positionierungen mittels NotesRichtextNavigator
- Wichtige Methode: FindAndReplace
- ACHTUNG: Wirft Zeilenumbrüche aus dem ersetzten Text, Workaround nötig!
- „Seltsame“ Definition von „TextRun“ und „TextParagraph“
- TextParagraph: Alles bis zum nächsten Text- Paragraphen
- TextRun: Alles bis zum nächsten Style- Wechsel (oder Paragraph, je nachdem was vorher kommt)

Properties

Navigator

Style

TextParagraph

TextRun

Type

Methods

Clone

FindAndReplace

Remove

Reset

SetBegin

SetEnd

SetStyle

Richtextmanipulation in LotusScript

Die NotesRichtextRange- Klasse (1/2)

- Kleiner Exkurs: Wie komme ich an die Platzhalter…

```
strInput = rtItem.Text

varTemp = Split( strInput , "{" )

ForAll strTemp In varTemp
  If strTemp <> "" Then
    strPlaceHolderName = StrLeft( strTemp , "}" )
    If strPlaceHolderName <> "" Then
      strReplaceFrom = "{" & strPlaceHolderName & "}"
      '- Hier die Funktion der Wahl einfügen, um die Platzhalterwerte zu ermitteln
      strReplaceTo = GetKeywordAdvanced(strPlaceHolderName,"",KEY_TYP_IMPLODE,KEY_RET_STRING, "~", False, "" )
      If strReplaceTo = "" Then strReplaceTo = PH_EMPTY
      lstrFindAndReplace( strReplaceFrom ) = strReplaceTo
    End If
  End If
End ForAll
```

Richtextmanipulation in LotusScript

Die NotesRichtextRange- Klasse (2/2)

```
Set rtRange = rtItem.Createrange()
ForAll strReplace In lstrFindAndReplace
  strReplaceTo = CStr( strReplace )
  strReplaceFrom = ListTag( strReplace )
  intCurReplace = rtRange.Findandreplace(strReplaceFrom, strReplaceTo, RT_REPL_ALL + RT_FIND_CASEINSENSITIVE)
  intReplaceCount = intReplaceCount + intCurReplace
End ForAll

'- Dieser Stunt ist nötig, weil FindAndReplace alle NewLines (im ersetzen Text, nicht im Richtext) wegschmeisst
'- Dieser Code ist HÄSSLICH, aber war mal so notwendig in Version ??? und wurde nie mehr angefasst...
Set rtNav = rtItem.CreateNavigator
While rtNav.FindFirstString("~")
  Set rtRange = rtItem.CreateRange
  Call rtRange.SetBegin(rtNav)
  Call rtRange.SetEnd(rtNav)
  Call rtRange.Remove
  Call rtItem.BeginInsert(rtNav)
  Call rtItem.AddNewline(1)
  Call rtItem.EndInsert
  Call rtItem.Update()
'- Navigator vom neuen riItem neu generieren
  Set rtNav = rtItem.CreateNavigator
Wend
```

Richtextmanipulation in LotusScript

Die NotesRichtextTable- Klasse

- Kann für tabellarischen Aufbau in Richtextitems benutzt werden
- Formatierung im Default „AutoFit“, für manuelle Formatierung: Array mit NotesRichtextParagraphStyle
- Sehr viel Code für sehr wenig Output
- Sehr aufwändig

Properties

AlternateColor

Color

ColumnCount

RightToLeft

RowCount

RowLabels

Style

Methods

AddRow

Remove

RemoveRow

SetAlternateColor

SetColor

Richtextmanipulation in LotusScript

Die NotesRichtextTable- Klasse (1/2)

```
For i = 0 To 1
  Set rtStyles(i) = g_ses.CreateRichTextParagraphStyle
  rtStyles(i).LeftMargin = 0
 rtStyles(i).FirstLineLeftMargin = 0
 If i = 0 Then
  rtStyles(i).RightMargin = RULER_ONE_CENTIMETER * 1.5
 Else
  rtStyles(i).RightMargin = RULER_ONE_CENTIMETER * 5
 End if
Next

'- Anzahl Zeilen auslesen
intRows = UBound( doc.Getitemvalue( "Categories" ) ) + 2

Call rtItem.AppendTable (intRows, 2,, RULER_ONE_INCH, rtStyles)

'- Tabelle wieder auslesen
Set rtNav = rtItem.Createnavigator()
'- eigentlich müsste da ein If drum... aber die Tabelle wurde ja gerade erst erzeugt
Call rtNav.Findfirstelement(RTELEM_TYPE_TABLE)

'- Weitere Eigenschaften setzen
Set rtTable = rtNav.Getelement()
rtTable.Style = TABLESTYLE_ALTERNATINGROWS
Set color = g_ses.Createcolorobject()

color.NotesColor = COLOR_WHITE
Call rtTable.SetColor( color )
color.NotesColor = COLOR_LIGHT_GRAY
Call rtTable.SetAlternatecolor( color )
```

Richtextmanipulation in LotusScript

Die NotesRichtextTable- Klasse (2/2)

```
'- und jetzt die Werte schreiben
Call rtnav.FindFirstElement(RTELEM_TYPE_TABLECELL)
'- zuerst die Header
For intColumn = 1 To 2 Step 1
  Call rtItem.Begininsert(rtNav)
  Call rtItem.Appendtext( strTitles(intColumn) )
  Call rtItem.EndInsert
  Call rtnav.FindNextElement(RTELEM_TYPE_TABLECELL)
Next

intRow = 1

'- und jetzt die Zeilen
ForAll strCategory In doc.Getitemvalue( "Categories" )
 If strCategory <> "" Then
  Call rtItem.Begininsert(rtNav)
  Call rtItem.Appendtext( CStr( intRow ) )
  Call rtItem.EndInsert
  Call rtnav.FindNextElement(RTELEM_TYPE_TABLECELL)
  Call rtItem.Begininsert(rtNav)
  Call rtItem.Appendtext( CStr( strCategory ) )
  Call rtItem.EndInsert
  Call rtnav.FindNextElement(RTELEM_TYPE_TABLECELL)
  intRow = intRow + 1
 End If
End ForAll
```

Richtextmanipulation in LotusScript

Arbeiten mit Hilfsmasken / Hilfsdokumenten

Hilfsmasken

- Sehr freie Gestaltungsmöglichkeiten
- Verwendung von Feldern / Formeln
- Keine Manipulation durch Anwender
- Starr, nicht / sehr eingeschränkt konfigurierbar

Hilfsdokumente

- Weniger Gestaltungsmöglichkeiten
- Manipulation durch Anwender möglich
- Jederzeit ohne Design- Anpassung änderbar

Richtextmanipulation in LotusScript

Arbeiten mit einer Hilfsmaske

```
Set doc = g_dbCurrent.Unprocesseddocuments.Getfirstdocument()
Set rtItem = doc.GetFirstItem( "Body" )

'- temporäres Dokument erstellen
Set docTemp = New NotesDocument( g_dbCurrent )

'- Header rendern
Call docTemp.Replaceitemvalue( "Form", "DemoTableHeader" )
Call docTemp.Rendertortitem( rtItem )

intRow = 1

'- Zeilen Rendern
ForAll strCategory In doc.Getitemvalue( "Categories" )
  If strCategory <> "" Then
    If intRow Mod 2 = 0 Then
      strForm = "DemoTableRowEven"
    Else
      strForm = "DemoTableRowOdd"
    End If
    Call docTemp.Replaceitemvalue( "Form", strForm )
    Call docTemp.Replaceitemvalue( "Pos", intRow )
    Call docTemp.Replaceitemvalue( "Category", strCategory )
    Call docTemp.Rendertortitem( rtItem )
    intRow = intRow + 1
  End If
End ForAll

Call rtItem.Update()

Call ws.Editdocument( True, doc )
```

Richtextmanipulation in LotusScript

Arbeiten mit einem Hilfsdokument

```
Dim ws As New NotesUIWorkspace

Dim doc As NotesDocument
Dim rtItem As NotesRichTextItem

Dim docHelper As NotesDocument
Dim rtHelper As NotesRichtextItem

'- Helper holen
Set docHelper = g_dbCurrent.Unprocesseddocuments.Getfirstdocument()
Set rtHelper = docHelper.Getfirstitem( "Body" )

'- Neues Dokument kreieren
Set doc = New NotesDocument( g_dbCurrent )
Call doc.Replaceitemvalue( "Form", "defaultForm" )
Set rtItem = New NotesRichTextItem( doc, "Body" )

'- Helper einfach anhängen
Call rtItem.Appendrtitem( rtHelper )

'- Hier könnte wieder der Code für die Platzhalter stehen, etc...

Call rtItem.Update()

Call ws.Editdocument( True, doc )
```

Richtextmanipulation in LotusScript

Backend vs. Frontend

- Bisher alle Beispiele im Backend… *gähn*
- Wie macht man solche Manipulationen im Frontend sichtbar
- Einfachster Fall: Im Backend vorbereiten, dann ins Frontend ziehen
- Bis vor kurzem „geheimer“ Befehl aus Mailtemplate: NotesUIDocument.ImportItem
- Geht aber nur in bearbeitbaren Richtextfeldern

- Wenn man aber das aktuelle Dokument aktualisieren will bzw. mit berechneten Items arbeitet – nur reopen !

Richtextmanipulation in LotusScript

Backend vs. Frontend

- Letzte Möglichkeit (wieder nur in bearbeitbaren Feldern)
- Template- Document (oder temporäres Dokument) kurz zum Bearbeiten öffnen
- NotesUIDocument- Methodern verwenden:
  - Per .GotoField ins Richtextfeld
  - Per .SelectAll alles markieren
  - Per .Copy in die zwischenablage
  - Mittels .Close das Dokument sofort wieder schliessen (User sieht maximal ein kurzes flackern)
  - Mittels .GotoField im „ursprünglichen Dokument“ ins Body- Feld wechseln
  - Optional: mit .SelectAll alles selektieren, um den Inhalt zu überschreiben
  - Per .Paste den Inhalt einfügen

Richtextmanipulation in LotusScript

Backend vs. Frontend

```
Dim ws As New NotesUIWorkspace

Dim uidoc As NotesUIDocument

Dim viwTemplates As NotesView
Dim docTemplate As NotesDocument

'- View setzen
Set viwTemplates = g_dbCurrent.Getview(
"VwLkpTemplate" )
Set docTemplate =
viwTemplates.Getfirstdocument()

'- ui- variable setzen
Set uidoc = ws.Currentdocument

Call uidoc.Gotofield( "Body" )
Call uidoc.Importitem( docTemplate, "Body" )
```

```
Dim uidoc As NotesUIDocument
Dim doc As NotesDocument
Dim rtItem As NotesRichTextItem

Dim viwTemplates As NotesView
Dim docTemplate As NotesDocument
Dim rtTemplate As NotesRichtextItem

Dim strUnid As String

'- View setzen
Set viwTemplates = g_dbCurrent.Getview( "VwLkpTemplate" )
Set docTemplate = viwTemplates.Getfirstdocument()
Set rtTemplate = docTemplate.Getfirstitem( "Body" )

'- ui- variable setzen
Set uidoc = ws.Currentdocument
Set doc = uidoc.Document
Set rtItem = doc.Getfirstitem( "Body" )

'- doc im backend updaten
Call rtItem.Appendrtitem( rtTemplate )

'- doc speichern
Call doc.Save( True, True, True )
'- unid merken
strUnid = doc.Universalid

'- damit das ui geschlossen werden kann: alle Referenzen
deleten
Delete doc
'- ui schliessen
Call uidoc.Close( True)

'- und wieder öffnen im Editmode
Call ws.Editdocument( True, doc )
```

Richtextmanipulation in LotusScript

HTML verwenden

- Einfachste Möglichkeit (nur Frontend):
  NotesUIDocument.Import(„HTML File“, strFilePath)

- Backend: sehr viel Code für Mime- Handling. Siehe Demo

Richtextmanipulation in LotusScript

Hotspots mit dynamischem Code versenden

- Wunsch- Dokument manuell erstellen und als DXL exportieren
- DXL extrahieren und parametrisieren
- Beispiel: „Dokumentenrumpf“

```
<?xml version='1.0' encoding='utf-8'?>
<!DOCTYPE profiledocument SYSTEM 'xmlschemas/domino_8_5_3.dtd'>
<profiledocument name='{BEFR.FORM}' {BEFR.USERNAME} xmlns='http://www.lotus.com/dxl'
 version='8.5' maintenanceversion='3.0' replicaid='{BEFR.REPLICAID}'>
<noteinfo noteid='{BEFR.NOTEID}' unid='{BEFR.UNID}' sequence='1'></noteinfo>
<item name='{BEFR.RTINAME}'><richtext>
<pardef id='1' leftmargin='1in'/>
{BEFR.RTICONTENT}</richtext></item>
</profiledocument>
```

- Beispiel Hotspot:

```
<par def='1'>
<actionhotspot {BEFR.HOTSPOTSTYLE}><code event='click'>
<formula>{BEFR.FORMULA}</formula>
</code>{BEFR.LINKNAME}</actionhotspot></par>
```

- Mittels einfacher Text- Operationen manipulieren, und reimportieren
- Weiter wie vorher

Zeit für Ihre Fragen.

Weitere Infos:
bechtle.com