Active X Kurzanleitung

Über Active X lässt sich eine leistungsfähige Verbindung mit Excel herstellen. Mittels ED kann Excel mithilfe von VBA®-Befehlen vollständig gesteuert werden. Obgleich die Verwendung von Active X sehr einfach ist, sollte sich der Benutzer über einige Schwierigkeiten und Ausnahmen im Klaren sein. Diese Kurzanleitung wurde erstellt, um Ihnen bei Ihrer ersten Active X-Verbindung behilflich zu sein.

Schritt 1: Eine Anwendung erstellen
Um Daten an eine Active X-Anwendung zu senden, muss ein Zeiger oder ein Handle auf diese Anwendung erstellt werden. Dies wird mit dem Befehl `DIM` bewerkstelligt.

```
Dim([ExcelApp], vbOle)
```

Es wurde nun eine Variable erstellt, welche das Handle auf die Excel®-Anwendung aufnehmen kann. Mit dem nächsten Befehl wird die Excel-Anwendung erstellt und der Variable zugewiesen.

```
ExcelApp := CreateOLE([Excel.Application])
```

Die nun erstellte Anwendung ist jedoch noch nicht sichtbar, außerdem enthält sie auch noch keine Arbeitsmappen. Es können jedoch VBA-Befehle an Excel gesendet werden.
Der 4DScript-Befehl zum Senden von Befehlen lautet `OLE`. Mit diesem Befehl können bis zu 4 Parameter übergeben werden. Im ersten Parameter wird das Handle angegeben, welches angesprochen werden soll, im zweiten der VBA-Befehl und im dritten der zu übergebende Wert.
Mit dem folgenden Befehl wird die Anwendung angezeigt.

```
Ole(ExcelApp, [Visible], 1)
```

In VBA würde dieser Befehl so lauten: '`Application.Visible = True`'. In ED steht das Handle '`ExcelApp`' für den Befehl '`Application.`', der dritte Parameter entspricht dem Wert.

Tip 1:
Um die verfügbaren Befehle einzusehen, wechseln Sie zu Excel und öffnen Sie die VBA-Umgebung über das Shortcut ALT +F11. Innerhalb der VBA-Umgebung wird dann mit der Taste F2 der 'Objektkatalog' gestartet. Wählen Sie im Feld 'Alle Bibliotheken' den Eintrag 'Excel'. Es werden nun alle verfügbaren Befehle angezeigt.

Schritt 2: Arbeitsmappe öffnen

Es kann nun eine neue Arbeitsmappe erstellt oder eine schon vorhandene geöffnet werden. In dieser Kurzanleitung werden beide Möglichkeiten vorgestellt.
Mit dem ersten Befehl wird eine Arbeitsmappe hinzugefügt und mit dem zweiten Befehl wird das aktive Blatt innerhalb der Arbeitsmappe gewählt.

```
Ole(ExcelApp, [Workbooks.Add])
Ole(ExcelApp, [ActiveSheet.Name], [Tabelle1])
```

Es wird nun eine Arbeitsmappe geöffnet.

Tip 2:
In VBA wird ein String mit Anführungszeichen umschlossen. Dies wird in ED mit den Zeichen [] bewerkstelligt. Somit müssen die Anführungszeichen mit [] Zeichen ersetzt werden.

Beispiel: `ActiveSheet.Cells(„A1:B4“)` wird zu `ActiveSheet.Cells([A1:B4])`

Tip 3:
Um in ED einen String anzugeben, muss dieser mit []- Zeichen umgeben werden. Es kommt vor, dass diese Zeichen als Teil des Codes übergeben werden müssen. Dafür stehen die Befehle sbo (Square bracket open) und sbc (Square bracket close) zur Vefügung.

Beispiel: `Concat([ActiveSheet.Cells(], sbo, [A1:B], String(Count), sbc, [)])`

Es wird nun angenommen, dass die Datei Beispieldatei.xls in C:\ vorhanden ist. Diese Datei kann nun folgendermaßen geöffnet werden:

```
Ole(ExcelApp, [Workbooks.Open([C:\\Beispieldatei.xls])])
```

Tip 4:
Das \-Zeichen hat eine andere Bedeutung in VBA. Um dieses Zeichen korrekt an Excel zu übergeben, muss es mit einem zusätzlichen \-Zeichen übergeben werden.
Aus diesem Grund ist die Verwendung des Befehls 'pdir' nur über einen Work-Around möglich. Die nachstehende Funktion übernimmt einen Pfad und gibt ihn mit doppelten Backslashes zurück.

```
RegisterFunction(
 [Xpdir],
 [Files],
 1, 1,
 [Do (
  Model.XString := [],
  Repeat(
   SubstrCount(p(1), [\]),
   Model.XString := Concat(
    Model.Xstring, StrSeperate(p(1), [\], Count), [\\]
   )
  ),
  Model.Xstring
 )
]
)
```

Falls die Datei 'Beispieldatei.xls' nicht im Pfad C:\ gespeichert ist, sondern im 'Work'-Verzeichnis von ED, würde der nachstehende Befehl die Datei öffnen. Der Vorteil ist, dass nun der Pfad relativ zum Installationsverzeichnis von ED angegeben werden kann.

```
Ole(
 ExcelApp,
 Concat(
  [Workbooks.Open(],
  Sbo, Concat(XpDir(Pdir), [Work\\Beispieldatei.xls]), Sbc,
 [)]
 )
)
```

Schritt 3: Die Excel-Datei bearbeiten

Alle Formatierungen, die direkt in Excel vorgenommen werden können, können auch direkt aus ED gemacht werden. Der nachfolgende Befehl ändert die Umrandung der angegebenen Zellen.

```
Ole(ExcelApp,
 [ActiveSheet.Range([A1:D1]).Borders(4).Weight], [3])
```

Tip 5:
Alle Konstanten haben in Excel einen aussagekräftigen Namen. Die untere Linie einer Zelle wird beispielsweise '`xlEdgeBottom`' genannt. Diese Konstanten werden als Werte übergeben. Um eine Übersicht über die in Excel vorhandenen Konstanten zu erhalten, öffnen Sie den 'Objektkatalog'. Dies wird in Tip 1 beschrieben.

Tip 6:
Aktionen können in Excel auch aufgenommen und in einem Makro gespeichert werden. Dies geschieht über den Eintrag 'Extras → Makro → Aufzeichnen'. Anschließend kann das Makro in ED übertragen werden.

Für weitere Hinweise zu VBA sei an dieser Stelle auf die Hilfe von Excel verwiesen.