



Midrange MAGAZIN

Aktuelle Ausgabe Juni 2005

Datenaustausch und -zugriffsarten

Excel, Word und die i5

Viele erfolgreiche Zusatzprogramme beschäftigen sich mit der Konvertierung von i5-Daten in Excel oder Word. In diesem Artikel finden sie Beispiele zum Programmieren eigener Anwendungen. Sie sehen, wie i5-Daten in MS-Office ausgegeben oder von dort eingelesen werden. Natürlich können Sie die Office-Programme auch über i5-Daten steuern. Unser Werkzeug ist Microsoft DotNet, wir verwenden die Sprachen VisualBasic von Microsoft und VisualRPG von ASNA, auf die i5 greifen wir über ODBC oder DataGate von ASNA zu.

Echte Office-Integration birgt ein großes Rationalisierungspotenzial

Sehr oft geht Zeit damit verloren, auf der i5 vorhandene Daten unter Windows herzustellen oder zu verarbeiten. Ein Paradebeispiel dafür ist die Verarbeitung von i5-Daten über die MS-Office-Produkte.

Ein praktikabler Lösungsansatz ist der Einsatz von DotNet-Programmen, die Daten von der i5 holen und in die MS-Office-Programme einbinden und diese – wenn nötig – auch steuern.

Dieser Artikel zeigt einige Anwendungsbeispiele aus der Praxis. Der Lösungsansatz wird beschrieben, Sie finden hier auch Details der Umsetzung. Den Programmcode für die Beispiele können Sie über den Autor beziehen. Senden Sie eine e-Mail an: c.neissl@niceware.at.

ODBC, OLEDB und Co.

Als DotNet-Programmierer ist man mit dieser Technologie vertraut. Im Hinblick auf die i5 hilft einem das aber relativ wenig. Die Möglichkeit, die i5 über die Standard-Technologie einzubinden, besteht zwar, sie hat aber Ihre Tücken – z. B. in der Performance. Um gelegentlich Daten aus der i5 zu holen, reicht es allemal. Als Basis für eine professionelle Anwendung ist dieser Weg nicht zu empfehlen.

Das erste Beispiel zeigt eine Übernahme von Lagerständen in eine Excel-Tabelle, wobei für jede Artikelgruppe ein Worksheet erzeugt wird.

WORD ist ein williges Opfer

Im zweiten Anwendungsbeispiel sehen Sie, wie ein Word-Dokument durch ein VisualRPG-Programm mit Daten vorbereitet und dem Anwender zum Fertigstellen übergeben wird. In früheren Zeiten wurde diese Aufgabenstellung über das inzwischen dahingeschiedene AS/400-Office-Vision gelöst.

Hier finden Sie einen zeitgemäßen Lösungsansatz, den Sie auch gleich auf Ihre Bedürfnisse abstimmen können. Um zu zeigen, wie vielseitig VisualRPG ist, habe

ich in diesem Projekt (quasi als Revanche für all die Beispiele, mit denen über ODBC usw. auf die i5 zugegriffen wird) die Datenbank „Access“ verwendet. Lassen Sie sich also von VisualRPG im Bezug auf Office-Automation überraschen.

Überall finden sich Miniatur-B2B-Lösungen,

die auf den Austausch von Excel-Daten beruhen. In unserem letzten Beispiel sehen Sie, wie eine TXT-Datei in Excel geladen wird, wie die darin enthaltenen Daten gegen die i5 geprüft werden und wie sie, wenn das „OK“ gegeben worden ist, übernommen werden. Wenn die Prüfung gegen die i5 nicht bestanden wurde oder sich Fehler in den Dateninhalten zeigten, wird eine entsprechende Meldung in Excel ausgegeben.

Die Sprache, in der diese Lösung erstellt wurde, ist VisualBasic. Der Zugriff auf die i5 erfolgt über DataGate von ASNA. Sehen Sie dazu Projekt 3.

Projekt 1 – Midrange-Lagerliste

Hier wird eine XLS-Datei mit Worksheets gefüllt, die die aktuellen Artikeldaten und Lagerstände enthält. Das Programm sucht zuerst die Artikelgruppen aus und erstellt für jede Artikelgruppe ein Worksheet innerhalb dieser Datei. In Abbildung 1 sehen Sie das Programm zur Laufzeit. In der Liste finden Sie die Artikelgruppen, die nacheinander abgearbeitet werden. Das Ergebnis sehen Sie in Abbildung 2.

Zum Zugriff wird eine ODBC-DataSource verlangt, die unter „Systemsteuerung _ Verwaltung _ Datenquellen“ angelegt werden muss. Danach kann über den Connection-String zugegriffen werden.

Das Programm ist parametergesteuert. In Abbildung 3 sehen Sie die Programm-Parameter. Die sind ausnahmsweise in einer INI-Datei abgelegt. Unter DotNet ist das ja out, man verwendet XML.

Abbildung 4 zeigt das Kernmodul, in dem das EXCEL-Objekt erstellt wird.

Außerdem wird das Erstellen des Worksheets für jede gefundene Artikelgruppe aufgerufen.



Projekt 1 – Abbildung 1

Microsoft Excel - Lagerliste.xls

	A	B	C
1	Artikelnummer	Artikelbezeichnung	Aktueller Lagerstand
2	30101	Biber-Rapid 1kg Schnellhärtender Fertigmörtel Agro	7
3	30102	Betonfix z. Ausbessern v. Beton 1kg	3
4	30103	Agroplan 8 Spachtelzement 25kg-Sack	0
5	30104	Estrilan K braun für Estriche 25kg-Kanister	0
6	30105	Biber-Rapid 5kg Schnellhärtender Fertigmörtel Agro	4
7	30106	Biber-Rapid 10kg Schnellhärtender Fertigmörtel	1
8	30107	Betonfix 5kg Haftemulsion für Beton	4
9	30112	Agro Waschbetonreiniger 1lt	2
10	30121	Avus Reiniger Öl- & Fettlöser 1lt	2
11	30129	Agrotex Schimmelkod 500ml	0
12	30130	Biber DM-F Betondichtmittel 5kg	1
13	30131	Biber DM-F Betondichtmittel 25kg	0
14	30132	Biber DM-F Betondichtmittel 1kg	2
15	30133	Agro Quellbandkleber Technicoll 8053 750g-Dose	0
16	30201	Klebefest Elastikkleber 500ml	0
17	30202	Klebefest Elastikkleber 1000ml	0
18	30210	RS 400 Rostschutz TOP 400ml	16
19	30211	SR 400 Scheibenklar 400ml	5
20	30212	MS400 Motorstarter 400ml	7

Navigation: H < > > > / BETON \ CHEM \ DECKE \ FARBE \ FB \ FE < >

Bereit

Projekt 1 – Abbildung 2 – Fertige Excel-Ausgabedatei

Lagerliste.ini - Editor

```

ConnString==Connection=ODBC;DSN=QDSN_ATWICE;UID=NE;PWD=XXX
SQLSelectDaten==SELECT * FROM MIDRANGE.ARTIKEL WHERE ARTGR = '#AGR#' ORDER BY ARTID
SQLSelectGruppen==SELECT DISTINCT(ARTGR) FROM MIDRANGE.ARTIKEL ORDER BY ARTGR
Platzhalter==#AGR#
FileName==Lagerliste.xls
HeaderTable==ARTID=Artikelnummer
HeaderTable==ARTBEZ=Artikelbezeichnung
HeaderTable==LAGER=Verfügbare Lagerstand

```

Projekt 1 – Abbildung 3 – Parameterdatei

```

oExcel = CreateObject("Excel.Application")
oBook = oExcel.Workbooks.Add
oExcel.Visible = False

iPbAnteil = Int(90 / lstGruppen.Items.Count) - 1
Dim dr As DataRow
For iNum = 0 To lstGruppen.Items.Count - 1
    stb.Text = "erstelle Sheet für Artikelgruppe " & lstGruppen.Items(iNum)
    CreateExcel(lstGruppen.Items(iNum), oBook, iNum + 1)
    pb.Value = pb.Value + iPbAnteil
Next iNum

pb.Value = 100
Me.Refresh()

oExcel.Visible = True

stb.Text = "Excel wird gespeichert in " + txtFileName.Text
Try
    oBook.SaveAs(FileName:=txtFileName.Text, CreateBackup:=False)
Catch ex As Exception
    MsgBox("Fehler bei speichern von " & Environment.NewLine _
        & txtFileName.Text, MsgBoxStyle.Exclamation, "Speichern des XLS")
End Try
oBook = Nothing

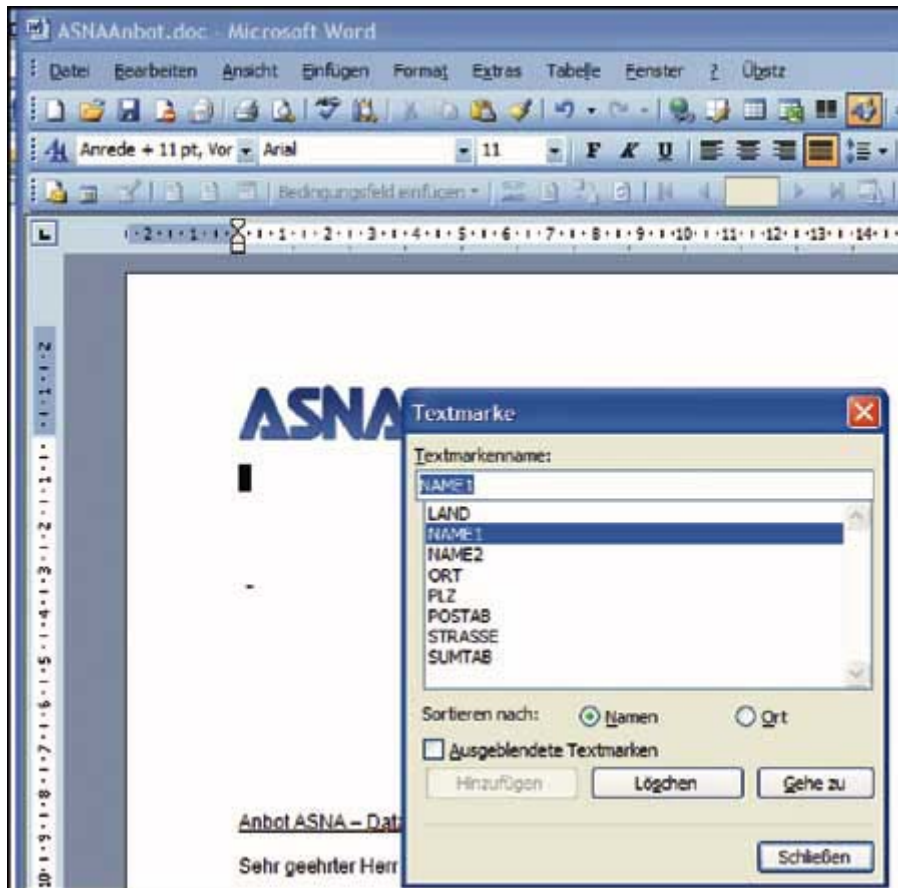
```

Projekt 1 – Abbildung 4 – Hauptschleife im Programm

Projekt 2 – Word-Automation

Dieses Modul stammt aus XMPL – meinem VisualRPG-OpenSource-Projekt. Im Word-Dokument sind Textmarken eingebaut, die die gleichen Namen wie die Datenfelder haben. Siehe Abbildung 1. In Abbildung 2 sehen Sie den Code, mit dem diese Textmarken angesprochen werden.

Woher weiß man nun, welche Funktion von den einigen tausend Funktionen, die Word anbietet, die richtige ist. Da hilft Word mit dem Makro-Editor weiter. Starten Sie den Editor und führen Sie die Schritte, die Sie von Word erwarten, manuell durch. Der Makro-Editor schreibt den Code, der dann nur noch in das Programm übernommen werden muss. Siehe Abbildung 4. Diese Erfolgsstrategie ist in allen Office-Produkten möglich.



Projekt 2 – Abbildung 1 – Textmarken im Formular

```

Selection          = WordApp.Selection

// Textmarke für jede Feld der Kopfdaten finden und Text einbauen
ForEach dc Type(DataColumn)  Collection(clsBK.DR.Table.Columns)

    Try

        Selection.GoTo(Word.wdGoToItem.wdGoToBookmark, *noparm, *noparm, dc.ColumnName)
        Selection.Find.ClearFormatting()
        Selection.TypeText(clsBK.DR.Item(dc.ColumnName.ToString()).ToString())

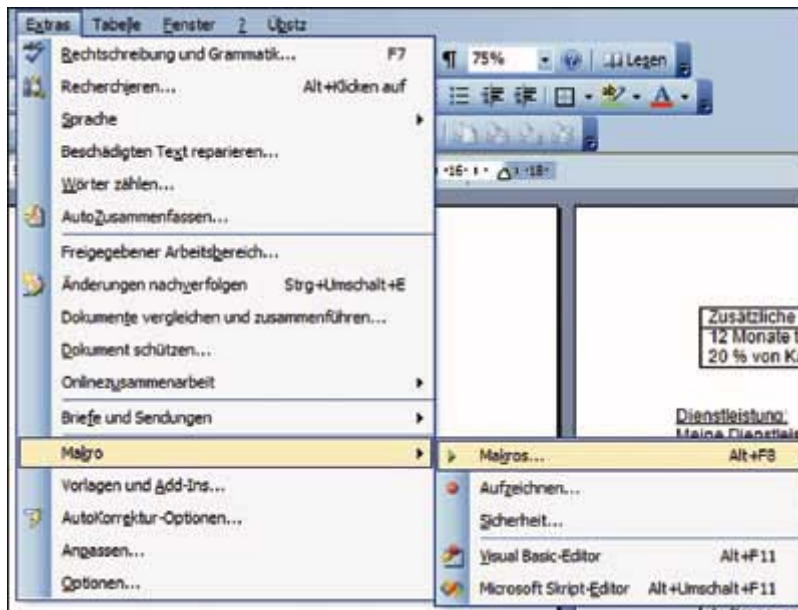
    Catch ex Exception
        MsgBox ex.Message

    EndTry

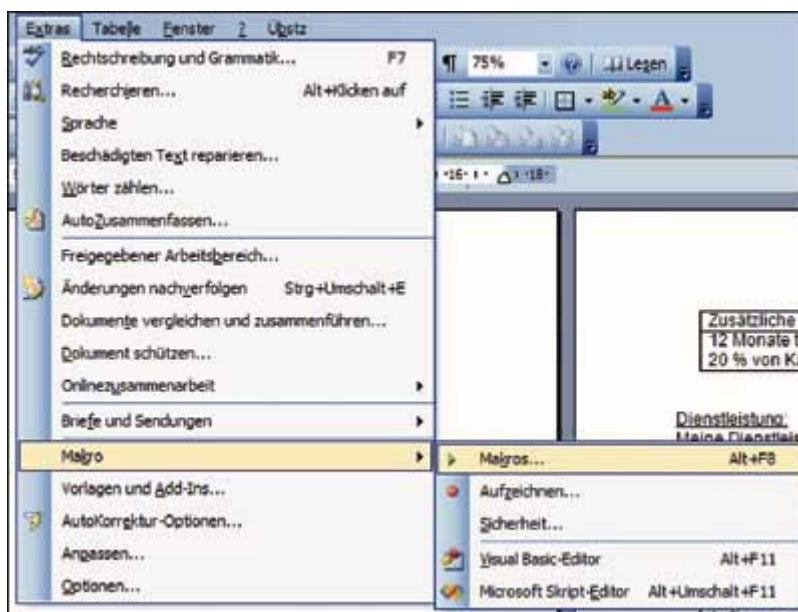
EndFor

```

Projekt 2 – Abbildung 2 – Ausfüllen der Textmarken



Projekt 2 – Abbildung 3 – Start des Word-Makro-Editors



Projekt 2 – Abbildung 4 – Word-Makro-Editor

Projekt 3 – EXCEL-MiniB2B-Lösung

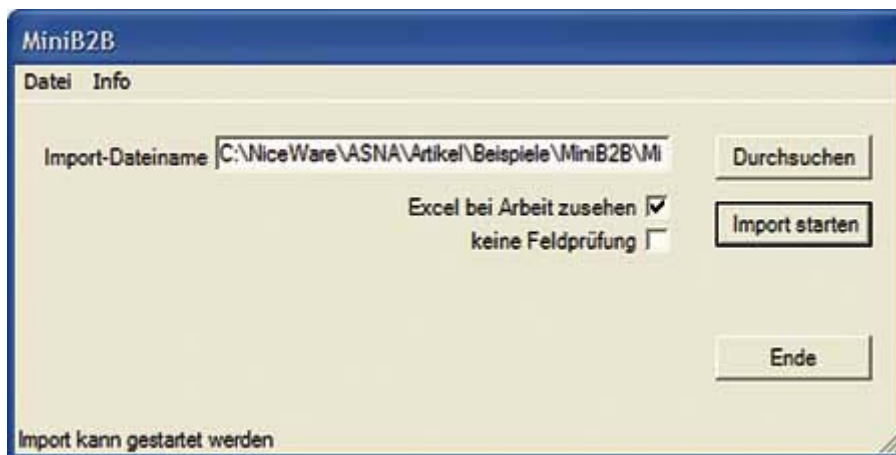
In diesem VisualBasic-Projekt wird demonstriert, wie über ASNA-DataGate i5-Tabellen angesprochen und aktualisiert werden können.

Unter VB stehen die selben Funktionen zur Verfügung wie in VisualRPG. Allerdings sind sie um einiges umständlicher zu programmieren, da nicht direkt auf Felder, sondern auf das DotNet-Standard-Datenobjekt (das DataSet) zugegriffen wird. Auch das hat seine Vorteile und kann unter Umständen durchaus sinnvoll sein.

In Abbildung 1 wird die Anwendung gezeigt, in der eine Excel-Tabelle bereits ausgewählt wurde. Mit der Checkbox „Excel bei der Arbeit zu sehen“ bekommt man während des Programmlaufs auch ein bisschen Unterhaltung geboten. In Abbildung 2 sehen Sie die fertig bearbeitete Tabelle, in der die fehlerhaften Felder markiert wurden.

In Abbildung 3 und 4 wird der Direktzugriff auf i5-Tabellen über DataGate demonstriert.

Abbildung 3 zeigt einen Zugriff über Satzschlüssel unter VB. Mit der Methode ReadRandomKey des FileAdapters wird auf den Satz zugegriffen, ohne eine Sperre auszulösen. Unter VisualRPG würde eine Zeile mit CHAIN genügen. Abbildung 4 zeigt einen Zugriff mit Sperre und ein Update. Beim Update wird der Datensatz zuerst mit PrepareRow instantiiert, dann durch die Funktion FillRecord mit Daten gefüllt und schließlich mit der Methode ChangeCurrent aktualisiert. Natürlich muss der Satz mit ReleaseCurrent wieder freigegeben werden. Für VisualRPG-Programmierer wäre das schlicht und ergreifend ein CHAIN mit folgendem UPDATE.



Projekt 3 – Abbildung 1 – Programmstart MiniB2B

	A	B	C	D	E	F	G	H
1	ARTIKEL	LIEFERANT	LZ	EKP	VVP	VON	BIS	(Feldnamen)
2	9001622000004	9003606600104	90	4.99	7.20	20050314	20050731	
3	9044400000001	9001650000007	12A	3.99	6.90	0	0	
4	9001622000004	9003606600104	90	5.99	9.90	20050801	20050731	(BIS-Datum < VON-Datum)
5	9001622000004	9003606600104	90	6.99	11.20	xxxx	xxxx	(BIS-Datum ungültig)

Projekt 3 – Abbildung 2 – Bearbeitete XLS-Tabelle

```
Private Function ChkEAN(ByVal iNr As Integer, ByVal strEAN As String) As Boolean
    ChkEAN = True
    If strEAN <> "" Then

        If iNr = 1 Then ' Lieferant

            dgLSKeyTable.Row(dbLSFld) = CLng(strEAN)
            Try
                dbLSFileAdapter.ReadRandomKey(daLSFileData, ReadRandomMode.Equal, LockRequest.NoLock, dgLSKeyTable)
            Catch dgEx1 As dgException
                ' wenn nicht gefunden
                If (dgEx1.Error = dgErrorNumber.dgEaNOTFND) Then
                    ChkEAN = False
                End If
            End Try
        End If
    End If
End Function
```

Projekt 3 – Abbildung 3 – Datenprüfung mittels Direktzugriff über Schlüssel auf i5-Datei

```

' Keylist bauen
dgKeyTable.Row(fnLFNR) = CLng(xlApp.Cells(1, 1).Value)
dgKeyTable.Row(fnMDNR) = CLng(xlApp.Cells(1, 2).Value)
If CStr(xlApp.Cells(1, 3).Value) = "" Then
    dgKeyTable.Row(fnBLNR) = 0
Else
    dgKeyTable.Row(fnBLNR) = CLng(xlApp.Cells(1, 3).Value)
End If

' verarbeiten wenn Key nicht leer
If dgKeyTable.Row(0) + dgKeyTable.Row(1) + dgKeyTable.Row(2) <> 0 Then

    Try

        ' Satz lesen
        dbFileAdapter.ReadRandomKey(dsFileData, ReadRandomMode.Equal, LockRequest.Default, dgKeyTable)

        ' Satz gefunden - wenn Update erwünscht
        If chkUpdate.CheckState = CheckState.Checked Then

            Try

                ' Satz updaten
                drFileRecord = dsFileData.PrepareRow(0)
                FillRecord(drFileRecord, 1)
                dbFileAdapter.ChangeCurrent(dsFileData)
                dbFileAdapter.ReleaseCurrent()

            Catch dgEx2 As dgException

                strMessage = "Update auf Datensatz nicht möglich - " + dgEx2.Message

            End Try

        Else

```

Projekt 3 – Abbildung 4 – Update eines i5-Datensatzes unter VB

Fachautor: Christian Neißl