



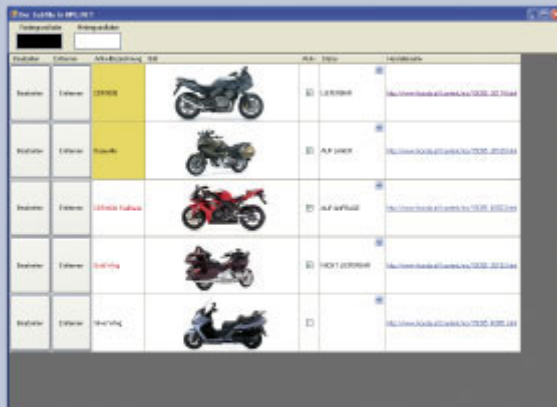
Ausgabe September 2006

Subfiles in .NET mit RPG.NET

Die grafische Benutzeroberfläche von Windows bietet jede Menge Möglichkeiten für die hübsche und funktionelle Aufbereitung von Anwendungen.

DataGrid ist ein Klassiker, der in VisualStudio 2005 neu gestaltet wurde. Nun bietet auch VS2005 im Standard Funktionalitäten, die bisher nur von 3rd-Party-Controls angeboten wurden. In diesem Beispiel beschäftigen wir uns mit den Möglichkeiten des DataGridView. Das neue Grid kann einige Dinge, die wir bisher vermisst haben – zum Beispiel: Bilder, Buttons, Dropdowns und Links in der Grid-Zeile darstellen oder Zellen nach Wunsch einfärben. Unser Beispiel in Abbildung 1 zeigt eine Übersicht, in der wir Grid und zwei Buttons für die Farbeinstellung finden. Per Mausklick wird die eingestellte Farbe im Grid übernommen. Die ersten beiden Spalten sind Buttons; die Artikelbezeichnung ist eine normale Textspalte. In der Bildspalte finden wir den Artikel dargestellt. Die Checkbox und das Dropdown zeigen den Artikelstatus; der Link verweist auf eine Internet-Seite.

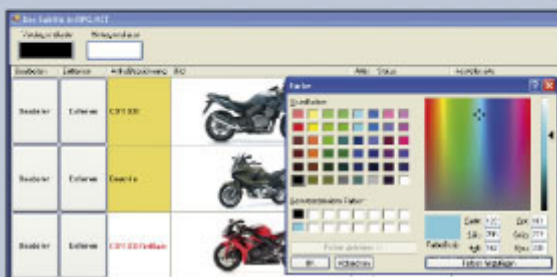
1 DATAGRID IN DER ANSICHT



2 DROPDOWN IN DER GRID-ZELLE



3 FARBAUSWAHL



Abhängig von der Einstellung können Daten im Grid auch bearbeitet werden; Abbildung 2 zeigt die Dropdown-Auswahl. Die Farbauswahl wird in Abbildung 3

gezeigt; nach Klick wird die Zelle wie gewünscht eingefärbt. Natürlich stammen die Daten von der iSeries; die Bilder werden von einem Netzwerk-Server oder lokal geladen.

4 PROGRAMMCODE LADEN DES SUBFILES

```
/// füllen der Subfile-Daten
BegSr LoadGrid

/// Verbindung zur DB aufbauen
Connect dglocal

/// Artikelstamm lesen
Read Stamm
/// Schleife über alle Artikel mit Ausgabe in den Subfile
Dowhile not %eof()
  Write memStamm
  Read Stamm
Enddo

/// Datentabelle aus dem Subfile für Grid vorbereiten
dt = memStamm.DataSet.Tables(0)

/// Verbindung zur Datenbank trennen
Disconnect dgLocal

/// Spalte für das Artikelbild als SYSTEM.OBJEKT erstellen
DclFld dcBild Type(DataColumn) new()
dcBild.ColumnName = „Bild“
dcBild.DataType = Type.GetType(„System.Object“)

/// Spalte für die Checkbox erstellen
DclFld dcAktiv Type(DataColumn) new()
dcAktiv.ColumnName = „Aktiv“
dcAktiv.DataType = Type.GetType(„System.Boolean“)

/// die Spalten an die Tabelle anfügen
dt.Columns.Add(dcBild)
dt.Columns.Add(dcAktiv)

/// für alle Sätze der Tabelle werden hier
/// die neuen Spalten mit Werten befüllen
ForEach dr Type(DataRow) Collection(dt.Rows)
  /// Checkbox mit Haken belegen
  dr.Item(„Aktiv“) = *True
  /// Bild aus JPG-Datei laden
  LOADPICTURE File(dr.Item(„IMSKU“) + „.jpg“) Target (dr.item(„Bild“))
EndFor

/// Subfile ans Grid übergeben
dgv.DataSource = dt

/// Farbeinstellungen für Auswahlbuttons vorbelegen
btnHintergrund.BackColor = Color.White
btnVordergrund.BackColor = Color.Black

EndSr
```

Verbindung zur iSeries aufbauen
und Artikeldaten einlesen

Spalten für Bilder und Checkbox
einbauen, die in der iSeries-
Datei nicht vorhanden sind

Laden des Bildes mit
RPG-OpCode
LOADPICTURE

Wie viel Code ist nötig?

Das ist auf Grund der Eigenintelligenz der Objekte sehr wenig. Für das Laden der Subfiles mit Daten und Bild kommt man – inklusive Kommentaren und vernünftig optischer Gestaltung des Codes – locker mit 50 Zeilen aus.

Am Beispiel des Farb-Dialogs (Abbildung 5) zeigt sich, dass man sich als Programmierer weder über den Dialog noch darüber Gedanken machen muss, ob die Benutzer diesen Dialog auch bedienen können. Durch Verwendung von Standards regelt sich das von selbst.

Die in Abbildung 6 dargestellte Routine wird beim Click-Ereignis ausgeführt. Hier befindet sich der Code für die MessageBox, der Aufruf der Internet-Seite und die Farbeinstellung der Grid-Zelle.

5 AUFRUFEN UND ABFRAGEN DES FARBDIALOGES

```
BegSr btnVordergrund_Click Access(*Private) Event(*this.btnVordergrund.Click)
Dc1SrParm sender Type(*Object)
Dc1SrParm e Type(System.EventArgs)

/// Farbdialog aufrufen und Farbe setzen
if clrDlg.ShowDialog() = DialogResult.OK
    btnVordergrund.BackColor = clrDlg.Color
Endif

EndSr
```

Hier wird der Windows-Standarddialog zur Farbeinstellung aufgerufen, bei OK wird die eingestellte Farbe am Button gespeichert.

6 CLICK-EVENT ABFRAGEN

```
/// Klick in Zelle abfangen
BegSr dgv_CellClick Access(*Private) Event(*this.dgv.CellClick)
Dc1SrParm sender Type(*Object)
Dc1SrParm e Type(System.Windows.Forms.DataGridViewCellEventArgs)

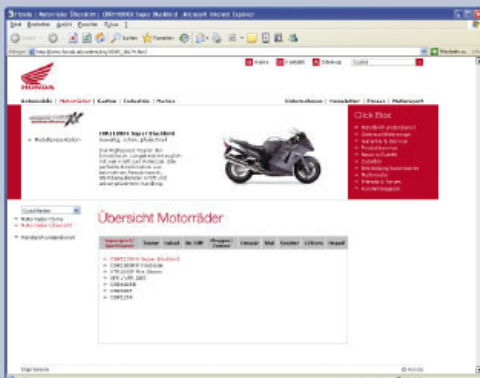
Dc1Fld strTxt *string
strTxt = „Wollen Sie Artikel „ + dgv.CurrentRow.Cells(„Artikel“).Value

Select
    /// wenn bearbeiten oder löschen gedrückt
    When dgv.CurrentRow.ColumnIndex = 0
        MsgBox Msg(strTxt + „ wirklich bearbeiten“) Icon(*question)+
        Title(„Button gedrückt“) Buttons(*yesno)
    When dgv.CurrentRow.ColumnIndex = 1
        MsgBox Msg(strTxt + „ wirklich Löschen“) Icon(*question) +
        Title(„Button gedrückt“) Buttons(*yesno)
    When dgv.CurrentRow.ColumnIndex = 6
        /// wenn Link gewählt einfach starten
        OSEXEC CMDLINE(dgv.CurrentRow.Value.ToString())
    Other
        /// sonst nur Farbe einstellen
        dgv.CurrentRow.Style.BackColor = btnHintergrund.BackColor
        dgv.CurrentRow.Style.ForeColor = btnVordergrund.BackColor
EndS1
EndSr
```

7 MESSAGEBOX



8 INTERNETSEITE



Noch ein Wort zu RPG

Wenn Sie sich nun fragen, wo denn da RPG ist, kann ich mit einer einfachen Antwort dienen: „Dieser Code ist RPG in seiner modernen Form.“

Moderne Software basiert auf Objekten. Wenn diese Software im Sinne des Erfinders entwickelt wird, hat sie auch das Ziel, selbst als Objekt zur Verfügung zu stehen. Im „klassischen“ RPG gibt es die hier dargestellten Möglichkeiten kaum, da sie von der Umgebung nicht geleistet werden können.

Es hat keinen Sinn an dieser Welt vorbeizugehen, da Sie ja „iSeries-RPG-Programmierer“ sind. Meiner Erfahrung nach als RPG.NET-Trainer ist bei den Entwicklern meistens die Angst vor der Hürde größer als die Hürde selbst. □