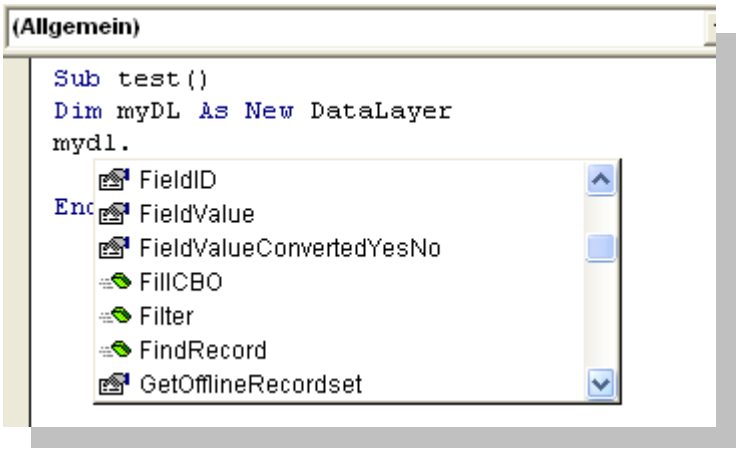
Entwicklerhandbuch
RedMill DataLayer 8.6



Die Benutzung des RedMill DataLayers
geschieht auf eigene Gefahr

Es wird jegliche Haftung abgelehnt.

Die in diesem Handbuch beschriebenen Verfahrensweisen
können ohne Meldung von RedMill geändert werden.
Aus diesem Handbuch entstehen für RedMill keinerlei
Verpflichtungen.

Rückmeldungen bezüglich der Verwendbarkeit
dieses Handbuches und der Software
sind sehr willkommen.

Neu in Version 8.6:

- Analyzer-Funktionen erweitert
- Diverse Verbesserungen

Inhaltsverzeichnis

Was ist der RedMill DataLayer?

Der RedMill DataLayer besteht hauptsächlich aus einer Com-Komponente (VB6) sowie verschiedenen Neben-Komponenten, welche eine Zwischenschicht darstellt für den Datenzugriff. Der DataLayer arbeitet mit Microsoft DAO Komponenten, Version 3.6 und kann daher für Microsoft Access-Datenbanken (ab 2000) eingesetzt werden.

Anstatt dass Sie in Ihrer Applikation an jeder Stelle, an der Sie Daten in die Datenbank schreiben oder aus der Datenbank lesen möchten, fix den Datenzugriff ausprogrammieren, rufen Sie einfach den DataLayer auf. Der DataLayer kümmert sich selber um das Lesen und Schreiben der Daten aus der Datenbank. Dabei bietet diese Datenzugriffsschicht verschiedene Möglichkeiten um Fehler zu vermeiden und um Fehler bereinigt zu erhalten.

Dies hat gewaltige Vorteile.

- Kein Problem mit NULL-Werten, denn diese werden direkt abgefangen und umgewandelt, dies müssen Sie nicht mehr an x-beliebigen Stellen selber tun.
- Umfangreicher Funktionsumfang: Suche nach Datensätzen, Filter und Sortierung benötigen nicht jedes Mal einen neuen Datenzugriff sondern können direkt erfolgen.
- Da die Daten in einer Zwischenschicht gespeichert sind, werden die Daten nicht sofort gespeichert, sondern erst auf Verlangen. Das Verwerfen der Änderungen ist daher möglich.
- Konnektoren in den RedMill DataLayerControlBar und RedMill DataGrid Steuerelementen erlauben es mit sehr wenig Programmieraufwand, direkt Daten aus dem DataLayer zu holen oder den DataLayer grafisch zu steuern. Einfache Datenverwaltungs-Formulare in Ihrer Applikation können im Handumdrehen erstellt werden.
- Daten können verändert werden, ohne dass sie gespeichert werden müssen.
- Der Datalayer kann selbstständig Datenbanken erzeugen und die Struktur anhand einer Versionskontrolle automatisch pflegen.
- Mit den integrierten Analyzer-Funktionen können Sie Berechnungen mit den Feldern anstellen, Daten auswerten und Pivots erstellen.

Gedanken zur Verwendung des RedMill DataLayers

Ich wollte einfach mal einen DataLayer quasi als Spass programmieren, ohne dass ich darauf angewiesen wäre. Ich habe schnell gemerkt, dass die Verwendung des DataLayers gewaltige Vorteile mit sich bringt. Nach einigen Jahren Entwicklungszeit, stellt der DataLayer heute ein mächtiges Werkzeug für die Handhabung aller Daten für sämtliche meiner Applikationen dar, egal ob die Daten in einer Datenbank gespeichert sind oder z.B. in einer XML Datei oder einfach nur im RAM. Ich erspare mir dadurch sehr viel Programmieraufwand, da ich mit wenigen Programmieranweisungen die Daten offline gehalten und visualisiert erhalte. Daraus resultiert ein Effizienzgewinn, da ich mich bei der Programmierung der Applikationen nicht mehr um die Datenhaltung kümmern muss.

Der Datalayer ist auch dann ein hervorragendes Mittel, wenn Datenbank-unabhängige Daten gehalten werden müssen. Beispielsweise wenn eine Applikation Werte in eine Datei speichern soll oder wenn für Berechnungen Zwischenresultate gespeichert werden sollen. Mit den Daten kann dann wie mit Daten in einer Datenbank gearbeitet werden. Die mitgelieferte XML-Komponente schreibt die Daten direkt in ein XLM File und liest diese wieder ein.

Auf der anderen Seite sollen die Nachteile nicht verschwiegen werden. Die Tatsache, dass die Daten offline kopiert und gehalten werden, hat einen grossen Einfluss auf die Performance und den Speicherbedarf. Übersteigt die Datenmenge den Speicher, erscheint die OVERFLOW-Fehlermeldung (das kommt allerdings höchst selten vor).

Der Datalayer eignet sich daher für folgende Einsatzgebiete:

- Einzelplatz-Applikationen
- Kleinere Netzwerk-Applikationen
- Dort wo eine Speicheranfrage nötig ist und nicht einfach direkt gespeichert werden soll
- Dort wo kleinere Datenmengen bearbeitet werden sollen
- Überall dort wo irgendwelche Daten datenbankähnlich bearbeitet werden sollen
- Wo Performance kein Kill-Kriterium ist.

In diesem Sinne wünscht Ihnen der Autor viel Spass bei der Programmierung.

Lizenzierung

Der RedMill DataLayer ist Freeware für nicht-kommerzielle Applikationen und unter der GNU GPL lizenziert, d.h. darf problemlos kopiert und weitergegeben werden. Nach einer 30-tägigen Testzeit erfolgt bei jeder Instanzierung eine Meldung an den Benutzer, dass die Komponente nicht lizenziert ist. Sie müssen einen Lizenzschlüssel setzen. Die Lizenz beträgt € 25.- pro kommerziellem Softwareprojekt, für welches Sie diese Komponenten einsetzen.

Für nicht-kommerzielle Softwareprojekte kann der DataLayer und die mitgelieferten Komponenten kostenlos verwendet werden.

So erhalten Sie Ihren Lizenzschlüssel:

1. Navigieren Sie auf http://software.redmill.ch
2. Navigieren Sie auf den Punkt RedMill DataLayer
3. Navigieren Sie auf den Punkt Lizenzschlüssel
4. Füllen Sie das Formular aus
5. Per E-Mail erhalten Sie automatisch den Lizenzschlüssel zugestellt (7 Tage/24h)

Es handelt sich also nicht um eine Registrierung. Das Absenden des Formulars löst automatisch das E-Mail mit dem Lizenzschlüssel aus. Ihre E-Mail Adresse wird vertraulich behandelt.

Source-Code

Eigentlich ist diese Komponente nicht als Open-Source gedacht. Auch ist der Code ziemlich umfangreich, sodass ich es mir erspare, diesen immer aktualisiert online zugänglich zu machen. Die Erfahrung zeigt, dass diesen eh kaum jemand herunterlädt. Daher ist der Code generell nicht öffentlich. Ich bin jedoch gerne bereit, Teile des Codes herauszugeben um bei der Problemlösung zu helfen.

Installation und Start

Der RedMill DataLayer ist eine Com-Komponente (VB6) und muss daher registriert werden. Wenn Sie ein Setup verwenden für Ihre Applikation, kann dieses Setup die Komponenten beim Endkunden installieren, ansonsten verwenden Sie die regsvr32 Funktion vom Betriebssystem. Der Start erfolgt über eine Instanzierung, entweder Early Binding oder Late Binding.

Mit Early Binding setzen Sie eine Referenz auf die Komponente (Nachteil bei Versionswechseln der Komponente muss die Referenz neu gesetzt werden) und arbeiten, wie wenn die Klassen in Ihrem Projekt selber wären. Mit Late Binding verwenden Sie die CreateObject Methode (RedMillDataLayer.DataLayer, u.s.w).

Es wird empfohlen, am Anfang mit EarlyBinding zu arbeiten, da das Codieren viel besser und schneller von der Hand geht. Sie können natürlich jederzeit auf Late Binding umstellen. Die verwendete Technologie ist VisualBasic 6.0. Inwiefern die Komponenten in anderen Sprachen funktionieren ist nicht bekannt.
Es ist empfohlen, dass Sie eine globale Funktion schreiben, welche das Objekt lädt, wie folgendes Beispiel zeigt:

```
 Function GetDataLayer() As Object
      On Error GoTo fehler
      Set GetDataLayer = CreateObject("RedMillDataLayer.DataLayer")
      GetDataLayer.SetDatabaseByPath=”c:\test.mdb”
      GetDataLayer.SetLicence “lizenzschlüssel”
      GetDataLayer.DebugMode=False
      Exit Function
fehler:
      MsgBox "Fehler beim Starten der Komponente RedMill DataLayer!" &
      vbNewLine & "Ev. könnte die Software nicht korrekt installiert
      sein.", vbCritical, "DataLayer"
End Function
```

Wenn Sie dann die Komponente instanzieren möchten, können Sie wie folgt vorgehen:

```
Dim objDatalayer As Object
Set objDatalayer = GetDataLayer()
```

Dies hat der Vorteil, dass Sie an einem Ort die Komponente instanzieren und allfällige Fehler abfangen können.

Hinweise zu dieser Dokumentation

Der RedMill DataLayer ist in VB6 geschrieben. Code-Beispiele werden daher auch in VB6 aufgeführt.

Inventar

Die folgenden Komponenten sind Teil des Lieferumfangs:

- RedMillDataLayer.dll
- RedMillXMLHandler.dll
- RedMillLogManager.dll
- RedMillFileManager.dll
- RedMillConfigurationManager.dll
- RedMillDataGridControl.ocx
- RedMillDataLayerControl.ocx
- RedMillFormDataControl.ocx
- Testprogramm.zip

Klassen

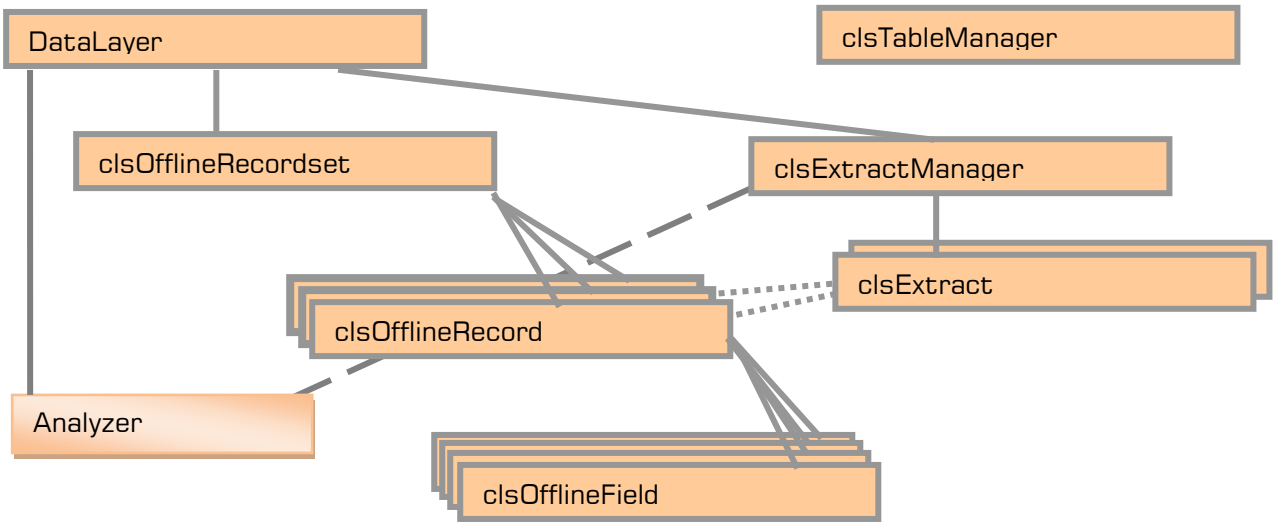
Alle Klassen, welche mit cls beginnen sollten in der Regel nicht selbständig instanziert werden, es sei denn, es ist in dieser Dokumentation beschrieben. .

Wichtige Regel

Der DataLayer schreibt Ihnen eine einzige Regel vor, die Sie jedoch zwingend befolgen müssen: In jeder Abfrage und in jeder Tabelle, die mit dem DataLayer bearbeitet wird, muss zwingend das Feld "ID" vorhanden sein, welches einen eindeutigen Wert beinhaltet, da sämtliche Speicher- und Löschaktionen sowie weitere Aktionen über die ID erfolgen.

Die Komponente wurde ausschliesslich unter Beachtung dieser Regel getestet. Wird diese Regel missachtet, kann es zu Fehlern kommen.

Objektmodell RedMillDataLayer.dll



Die Datalayer-Klasse ist die Hauptklasse, welche Sie für die Arbeit verwenden. Die Datalayer-Klasse hat sämtliche Eigenschaften und Methoden zur Verfügung, welche Sie für das Lesen und Speichern von Daten sowie für deren Handling benötigen.

Die Datalayer-Klasse instanziert die clsOfflineRecordset Klasse. Diese Klasse beinhaltet die virtuelle Tabelle mit den Daten. Sie können die clsOfflineRecordset Klasse selber instanzieren, wenn Sie für irgendwelche Zwecke eine virtuelle Tabelle benötigen, sie können das befüllte Objekt jedoch auch über den DataLayer ansprechen und auch auskoppeln.

Das clsOfflineRecordset beinhaltet eine Collection, welche die clsOfflineRecord Objekte beinhaltet. Dies stellt die Datensätze dar. Jeder Datensatz (clsOfflineRecord Objekt) führt wiederum eine Collection, in der die clsOfflineField Objekte gespeichert sind.

Das clsTableManager Objekt beinhaltet Methoden zur Erstellung/Bearbeitung von Tabellen und Abfragen (DAO). Sie können damit neue Tabellen oder Abfragen in der Datenbank erzeugen, Felder hinzufügen, u.s.w.
Die Klasse clsFilter wird nur intern benötigt

Excel-Klasse

Zusätzlich zu dem obigen Objektmodell gibt es noch die ExcelTools Klasse, welche Funktionen für Microsoft Excel zur Verfügung stellt. Diese Klasse kann direkt aus dem Datalayer instanziert werden, indem die Excel-Property verwendet wird. Dabei werden die Daten aus dem Datalayer direkt in ein Workbook übertragen, das über die ExcelTools-Klasse gesteuert oder abgeholt werden kann.

Arbeit mit Access-Datenbanken

Wenn Sie den DataLayer in Verbindung mit einer Access-Datenbank benutzen, müssen Sie zuerst dem DataLayer mitteilen, mit welcher Datenbank gearbeitet werden soll. Dafür stehen verschiede Methoden zur Verfügung. Entweder übergeben Sie ein DAO Objekt, ein Recordset-Objekt oder den Pfad zur Datenbank (alternativ mit Workgroup-Information).

Anschliessend können Sie ein Query ausführen. Mit `GetRecordset("Select * FROM....")`, resp. mit dem von Ihnen gewünschten Datenbank-Query wird automatisch die virtuelle Tabelle erzeugt und die Daten werden aus der Datenbank in die virtuelle Tabelle kopiert.

Achtung! Update und Delete-Queries können in der clsTableManager Klasse ausgeführt werden, jedoch nicht im DataLayer selber.

Sie navigieren ähnlich wie bei DAO in einem Recordset-Objekt von Datensatz zu Datensatz mit MoveFirst (erster Datensatz), MoveLast (letzter Datensatz) und MoveNext/MovePrevious (nächster, vorhergehender Datensatz). Mit MoveSpecific können Sie den Cursor auf die gewünschte Position des Datensatzes innerhalb des Recordset-Objektes setzen..

Sie können mit `strMeinWert = FieldValue("MeinWert")` Daten auslesen und mit `SetFieldValue "MeinWert", strMeinwert` einen neuen Wert setzen.

Wichtig: Die Werte werden nicht automatisch in der Datenbank abgespeichert. Es muss explizit die SAVE-Methode aufgerufen werden.

Arbeit ohne Access-Datenbanken

Der RedMill DataLayer ist ein hervorragendes Instrument um irgendwelche Daten strukturiert und datenbankähnlich zu halten und zu bewirtschaften. Dabei stehen die gleichen Eigenschaften und Methoden zur Verfügung wie bei der Arbeit mit Datenbanken. Einziger Unterschied ist:
Es kann die Save-Methode nicht verwendet werden. Sie müssen die Daten z.B: in eine XML-Datei abspeichern. Dafür steht eine entsprechende Methode zur Verfügung.

Eine neue Tabelle kann entweder direkt im OfflineRecordset angelegt werden oder mit der Methode `CreateNewVirtualOfflineTable`. Dabei müssen die Feldnamen übergeben werden.

Sie können jederzeit auch direkt auf die virtuelle Tabelle (clsOfflineRecordset) zugreifen oder gänzlich nur alleine mit der virtuellen Tabelle arbeiten, ohne DataLayer.

ErrorStatus

Der DataLayer gibt nur dann eine Fehlermeldung aus oder erzeugt einen Fehler, wenn der Fehler kritisch ist, z.B. wenn eine Datenbankabfrage nicht funktioniert. In allen anderen Fällen wird der DataLayer keine Fehlermeldung direkt an den Benutzer ausgeben. Wenn etwas nicht funktioniert, wie Sie es sich wünschen, dann prüfen Sie doch den ErrorStatus. Diese Eigenschaft ist weiter hinten in dieser Dokumentation im Detail beschrieben. Zudem können Sie den DebugMode auf True setzen (siehe weiter unten)

Die DataLayer-Klasse – Eigenschaften und Methoden

1. Administration

LogPath: Setzt oder gibt den Pfad für das automatische Erstellen eines Logs aus. Es findet ein automatisches Logging mit allen Änderungen statt. Sie können dieses Log verwenden um z.B. Aktionen rückgängig zu machen. Als Standard wird der Pfad der Komponente genommen. Das Log wird im XML-Format geschrieben. Es erfolgt keine Löschung alter Logfiles, dafür gibt es eine Methode in der RedMill LogManager Komponente, welche aufgerufen werden kann.

DisableLog: Default False. Mit True wird kein Log mehr geschrieben (=Performancegewinn).

Username: Setzt den Benutzernamen. Dieser wird für das Log verwendet.

SetDatabase: Wenn Sie bereits in Ihrem Projekt eine Datenbank geöffnet haben, können Sie diese mit dieser Eigenschaft übergeben. Dann arbeitet der Datalayer mit dieser Datenbank.

SetDatabaseByPath:
Öffnet die Datenbank mit dem angegebenen Pfad

SetDatabaseByPathAndWorkgroup
Wenn Ihre Datenbank mit einer Workgroup geschützt ist, können Sie die Datenbank über diese Funktion öffnen

SetStandardDBPathFile
Mit dieser Methode können Sie z.B. in Ihrem Projekt die Klasse einmal beim Start aufrufen und den Standardpfad übergeben. Dadurch ersparen Sie es sich, bei jeder Instanzierung die obigen Eigenschaften und Methoden anzuwenden. Bei dieser Methode können Sie auch das Workgroupfile sowie Username/Password angeben.
Sie sollten diese Methode nur verwenden, wenn Sie den DataLayer für ein einziges Projekt einsetzen. Besser ist es, eine zentrale Funktion zu schreiben, welche den Datenbank-Pfad automatisch bei der Instanzierung speichert (siehe Seite 4)

SetLicence: Am besten instanzieren Sie beim Start Ihres Programms die Komponente einmal und übergeben nicht nur den Standardpfad wie oben sondern auch den Lizenzschlüssel.

CreateNewVirtualOfflineTable:
Wenn Sie ohne Datenbank arbeiten können Sie eine neue Offline-Tabelle anlegen und die Feldnamen übergeben. Dann können Sie mit AddNew neue Datensätze hinzufügen und diese datenbankähnlich bewirtschaften.

- SetAutoGuidColumn:
  Wenn Sie anstatt einer ID mit einer GUID arbeiten möchten, können Sie mit dieser Methode bestimmen, dass der DataLayer sich autonom um die Bestückung des Felds mit Guids gemäss übergebebenen Feld-Parameter kümmert. Beispielsweise können Sie eine Spalte „Guid“ erstellen und diese als Auto-Guid Feld deklarieren. Neue Datensätze werden so direkt mit einer GUID im angegebenen Feld bestückt. Zudem kann mit dem Parameter bolUpdateVirtualTableNow=true alle GUIDS sofort gesetzt werden. Der Algorithmus wird jedoch nur dort GUIDS setzen, wo das Feld noch leer ist.

- AddVirtualField: Mit dieser Methode können Sie ein Feld an der virtuellen Tabelle anhängen. Beachten Sie bitte, dass das Feld nicht in der Datenbank vorkommt! Sie dürfen diese Methode daher nur für virtuelle Tabellen ohne Datenbank verwenden. Ansonsten erstellen Sie bitte zuerst das Feld in der Datenbank mit dem TableManager. Mit der Übergabe des Feldnamens wird das neue Feld in der virtuellen Tabelle erstellt.

- DebugMode: Wenn dieser Schalter auf Wahr gesetzt wird, gibt der DataLayer beim Beenden des Objekts eine Meldung mit dem ErrorStatus aus.

- ErrorStatus: Über diese Eigenschaft kann der Errorstatus abgefragt werden. Bei einem Fehler schreibt die Klasse die Fehlerbeschreibung in diese Eigenschaft. Löschen mit `.ErrorStatus=""`

NB: Die Methode „CreateAnewDatabase“ wird nicht mehr unterstützt und ist daher „deprecated“. Die Methode wird in einer nächsten Version verschwinden. Bitte verwenden Sie die Methode CreateDatabase in der clsTableManager Klasse.

2. Arbeit mit Daten

- GetRecordset: Dies ist die Hauptmethode des DataLayers. Damit können Sie eine beliebige SELECT SQL-Abfrage ausführen lassen. Die Daten befinden sich anschliessend in der erzeugten virtuellen Tabelle. Die SQL Abfrage kann mit SQLStringInUse abgefragt werden.

- MoveFirst/Last: Diese Methoden bewirken das gleiche wie bei einem DAO-Recordset. Sie können den Zeilen-Index zuoberst oder zuunterst in der virtuellen Tabelle setzen.

- MoveNext/Previous:
  Setzt den Zeilenindex eine Zeile nach unten oder eine Zeile nach oben. Achtung! Hier ist eine automatische Fehlerbehebung eingebaut, d.h. am Ende der Tabelle bewirkt MoveNext nichts und es erfolgt keine Fehlermeldung, wie auch bei MovePrevious.

- MoveSpecific: Damit können Sie den Index direkt steuern und auf eine bestimmte Zeile setzen (was bei DAO ja nicht geht)

- FillField: Mit dieser Methode können Sie ein Feld mit einem Wert beschreiben. Sie können dabei wählen, ob existierende Werte überschrieben werden sollen.

- ActualPosition: Mit dieser ReadOnly Eigenschaft können Sie den aktuellen Zeilenindex abfragen.

- FieldValue: (ReadOnly) Über die Angabe des Feldnamens können Sie den Wert des entsprechenden Felds der aktuellen Zeile abfragen.
  Beispiel: `strName = dloUsers.FieldValue("Name")`
  Die FieldValue Eigenschaft behandelt direkt die NULL-Werte aus der Datenbank, d.h. Sie müssen sich nicht mehr darum kümmern.

- FieldValueConvertedYesNo
  Diese Eigenschaft arbeitet wie die FieldValue-Eigenschaft, kann zusätzlich jedoch Booleische Werte in Ja/Nein umwandeln. Liegt kein True/False vor, dann wird der korrekte Wert ausgewiesen.

- SetFieldValue
  Mit dieser Eigenschaft können Zellwerte zurückgeschrieben werden. Beispiel:
  `dloUsers.SetFieldValue "Name", strName`

- Analyzer Gibt ein Analyzer-Objekt zurück

- AddNew: Diese Methode erzeugt einen neuen Datensatz in der virtuellen Tabelle. Nach dem Abspeichern mit der Save-Methode wird die ID sofort verfügbar sein.

- Save: Speichert sämtliche Änderungen in allen Datensätzen ab. Es ist sehr empfohlen, die Save Methode nicht für jeden einzelnen Datensatz anzuwenden sondern global am Ende der Verarbeitung (der Benutzer könnte so z.B. noch abbrechen und die Daten sind dann nicht gespeichert). Die Save-Methode macht alle ID's der neuen Datensätze (AddNew) sofort verfügbar.

- RemoveRecord: Mit der Angabe `bolDeleteAlsoInDB = True` wird der Datensatz auch direkt in der Datenbank gelöscht, ansonsten wird der Datensatz nur in der virtuellen Tabelle gelöscht. Wenn hier false angegeben wird, kann der Datensatz später nicht mehr nachträglich auch noch in der Datenbank gelöscht werden (es sei denn, es wird ein neues Datalayer-Objekt erzeugt).

- CopyCurrentRecord:
  Mit dieser Methode kann ein Datensatz kopiert werden. Dabei muss der Feldname der ID angegeben werden. Es kann auch eine neue ID übergeben werden, dann erhält der Record diese neue ID. Dies ist von Vorteil, wenn Sie den DataLayer ohne Datenbank verwenden, d.h. als virtuelle Datenhaltung. Bei der Verwendung mit Datenbank wird bei der Save-Methode die ID aus der Datenbank übernommen.

- RepairRecords: Stellt die Integrität der Records sicher bei der Arbeit ohne Datenbank (falls Felder verschoben sind oder Felder fehlen). Dies ist z.B. dann von Vorteil wenn Sie mit einem Daten-Import arbeiten und dabei die Feldnamen vom Import übernehmen. Wenn mehrere Imports in das gleiche Recordset gelesen werden sollen, kann eine Verschiebung der Felder auftreten. Diese Methode stellt alles richtig.

3. Arbeit mit XML-Dateien

Wie später in dieser Dokumentation beschrieben ist, wird eine XML Handler-Komponente mitgeliefert. Der DataLayer beinhaltet 2 Methoden um diesen XML Handler einfach anzusprechen und Daten abzuspeichern. Dabei wird der XML-Handler gekapselt und die entsprechenden Methoden aufgerufen. Sie können direkt über den XML Handler verfahren oder bequemer über diese Methoden im DataLayer:

- SaveAsXML: Diese Methode speichert das aktuelle OfflineRecordset Objekt in der angegebenen XML-Datei ab. Eine allenfalls bestehende XML Datei wird ohne Rückfrage überschrieben.
- LoadFromXML: Diese Methode lädt die angegebene XML-Datei und erstellt mit den Daten ein neues OfflineRecordset Objekt.

Detailliertere Informationen zu den Parametern finden Sie im Kapitel RedMill XML Handler

4. Informationen

- FieldID: Gibt den Feldindex (Achtung: von 1 angefangen!) des abgefragten Feldnamens zurück.
- ReadOnly: Wenn ein DAO-Recordset aufgrund der Komplexität nur noch Readonly ist, gibt diese Eigenschaft True zurück (Eigenschaft kann nur abgefragt und nicht gesetzt werden)
- RecordCount: Gibt die Anzahl der Datensätze der virtuellen Tabelle zurück.
- CheckContainsFieldName: Prüft, ob der angegebene Feldname im aktuellen Datensatz vorhanden ist.

5. Suchen, Sortieren und Filtern

- FindRecord: Setzt den Record-Index (Datensatz-Zeiger) auf den ersten Record, der im gewählten Feld den angegebenen Vergleichswert enthält. Gibt True oder False zurück, ob ein Datensatz gefunden wurde. Die Abfrage des Rückgabewerts ist empfohlen, da bei False der Curser auf Position 1 steht.

- FullTextSearch: Mit dieser Methode kann die Volltextsuche gestartet werden. Im Gegensatz zu FindRecord wird eine Hitliste in Form einer Collection zurückgegeben und es wird über alle Felder gesucht, nicht nur über ein Feld.
  Parameter:
  - strSearchString: Der Suchtext (Wörter mit dem Leerschlag abgetrennt werden nach der ODER Logik gesucht, d.h. es kann entweder das eine oder das andere zutreffen).
  - strLimitToFieldNames: Die Suche kann auf bestimmte Feldnamen (mehrere Feldnamen mit Semikolon ; abtrennen) beschränkt werden (Performance!) Ist der Parameter nicht angegeben, wird über alle Felder gesucht, ausser über das Feld ID und über das AutoGuid-Feld
  - strHitListFields: Ohne Angabe dieses Parameters wird eine Collection zurückgegeben mit den Record-ID's der zutreffenden Datensätze. Wird der Parameter übergeben (einzelne Felder mit Semicolon abtrennen) wird eine Liste mit den mit Semicolon abgetrennten Werten pro Hit-Datensatz in der Collection übergeben. Diese Werte können Sie danach z.B. in einer Liste darstellen (Sie müssen die Semicolons dann umwandeln). Es empfiehlt sich das Feld ID hier einzuschliessen, damit Sie diesen Schlüssel weiterverwenden können.

- Filter (*): Die Filter-Methode sichert die virtuelle Tabelle und löscht anschliessend alle Datensätze, welche den angegebenen Kriterien nicht entsprechen. Wenn Sie innerhalb dieser restlichen Datensätze nach einem weiteren Kriterium filtern möchten, müssen Sie bolAdditionalFilter auf True setzen, ansonsten vor der erneuten Filterung die Methode DeleteFilter aufgerufen wird. Die Option bolNoAddCollection verhindert, dass der Filter gespeichert wird. Dies hat eine Auswirkung auf die Refresh-Methode, welche den gespeicherten Filter automatisch auf die gerefreshten Daten anwendet. Bei einem Refresh würden bei gesetztem bolNoAddCollection=True der Filter nicht berücksichtigt. Die Filter-Methode hat keine Auswirkung auf die Datenbank. Denken Sie daran, dass der DataLayer nicht überprüft, ob Daten verändert wurden, d.h. Sie müssen allenfalls vor der Filter-Methode noch einen Save oder Refresh machen. **(*) Die Filter-Methode empfiehlt sich dann, wenn Sie effektiv gewisse Datensätze wegfiltern und nicht mehr damit arbeiten möchten. Ansonsten ist die Extrakt-Methode in jedem Fall die bessere und effizientere Methode.**

- DeleteFilter: Löscht den Filter, d.h. aktiviert die vor der Filterung gespeicherte virtuelle Tabelle. Dies bedeutet, dass bei geänderten Daten diese verloren gehen, es sei denn, sie werden vorgängig gesichert. Anschliessend kann bei geänderten Daten ein Refresh durchgeführt werden, damit die virtuelle Tabelle wieder auf dem neusten Stand ist. Bei Daten, welche eh nur abgefragt werden, erübrigt sich dieses (was bei der Filterung der Normalfall ist). Wenn Daten im Filter-Objekt geändert werden, müssen diese zuerst gespeichert werden, bevor die Delete-Filter Methode aufgerufen wird, da sonst die Änderungen verworfen werden.

- RemoveMultipleEntries: Diese Methode vergleicht sämtliche Datensätze auf dem angegebenen Feld, ob eine Übereinstimmung festgestellt wird. Wenn zwei oder mehr Felder in der gewählten Spalte den gleichen Wert beinhalten, bleibt der erste Datensatz bestehen und die anderen Datensätze werden gelöscht. Dies hat jedoch keine Auswirkung auf die Datenbank.

- TrimField: Es kann sein, dass bei einem CSV-Import viele Leerzeichen vorhanden sind. Diese können mit der VB-Funktion trim() entfernt werden. Die Methode TrimFields „rattert“ durch alle Datensätze durch und ersetzt für das angegebene Feld den aktuellen Wert mit dem getrimmten Wert, d.h. ohne Leerzeichen.

- Extract: Diese Eigenschaft gibt ein clsExtractManager Objekt zurück. Die Beschreibung zu dieser Klasse finden Sie weiter hinten in dieser Dokumentation. **Sie arbeiten besser und effizienter mit den Extrakten als mit der Filter-Methode.**

- CalcSum: Diese Methode liefert die Summe des gewählten Felds aller Datensätze.

- SortRecordset: Diese Methode sortiert die Datensätze in der virtuellen Tabelle nach dem gewählten Kriterium (Feldname) und zwar in aufsteigender Reihenfolge (bolAsc) oder in absteigender Reihenfolge (bolDesc). Mit Angabe, ob numerisch oder alphanumerisch sortiert wird. Ist irgendein Wert alphanumerisch, wird in jedem Fall alphanumerisch sortiert.

- SortRecordsetNextInside:
  Wenn innerhalb einer sortierten Tabelle nach einem anderen Kriterium nochmals sortiert werden soll, muss diese Methode aufgerufen werden, da die Methode SortRecordset bisherige Sortierungen ignoriert. SortRecordsetNextInside sortiert auf dem gewählten Feld, wenn die Werte im zuletzt gewählten Feld (strFieldNameLastSorted) die Werte übereinstimmen. Diese Methode kann nur im Anschluss an SortRecordset gewählt werden, dann aber beliebig viel. Wichtig ist, dass immer der Feldname angegeben wird, nachdem vorher sortiert wurde.

6. Informationen zur Speicherung, Löschung und Refresh

- AddForceField: Wenn in der Access oder anderer DAO Datenbank eine Datenbeziehung mit Aktualisierungs- oder Löschweitergabe besteht, kann es sein, dass Datensätze nicht abgespeichert werden. Der Grund liegt darin, dass zuerst ein neuer Datensatz mit der ID angelegt wird und dann der Datensatz erst befüllt wird. Mit AddForceField kann der Feldname hinterlegt werden, der auf jeden Fall beim Anlegen eines neuen Datensatzes in der Datenbank abgespeichert werden muss. Diese Methode kann x-mal mit jeweils anderen Feldern aufgerufen werden.
- ClearForceFields:
  Diese Methode löscht sämtliche hinterlegte Force-Fields (Felder, welche unbedingt abgespeichert werden müssen).

- Refresh: Daten können unter Umständen von anderen Benutzern geändert werden und der aktuelle Benutzer hat in der virtuellen Tabelle nicht die aktuellen Daten. Aus diesem Grund gibt es die Refresh Methode, welche die Daten aktualisiert. Mit bolSaveChanges geben Sie an, ob die Änderungen beim aktuellen Benutzer in der Datenbank gespeichert werden soll.

- RefreshTimer: Diese Write-Only Eigenschaft setzt den automatischen Refresh. Nach einem bestimmten Intervall wird die virtuelle Tabelle mit den Daten aus der Datenbank aufgefrischt.

- AlternativeSelectQueryForRemove:
  Die SQL-Abfrage, welche für die Befüllung der virtuellen Tabelle übergeben wurde, wird vom DataLayer auch für die Löschung von Datensätzen verwendet. Wenn das Query jedoch zu komplex ist, kann die Löschung nicht funktionieren. Aus diesem Grund kann über diese Write-Only Eigenschaft ein alternatives SELECT-Query übergeben werden, welches für die Löschung von Datensätzen dann verwendet wird.

Refresh

Nachdem die virtuelle Tabelle mit der Datenbank nicht verknüpft sondern "offline" ist, müssen Sie sich Gedanken machen, wie Sie in Ihrer Applikation den Refresh vornehmen möchten bei Daten, welche vom Benutzer praktisch immer offen gehalten werden. Am Beispiel einer Adressdatenbank wird diese Thematik erläutert:

Wenn mehrere Benutzer mit der Datenbank arbeiten und einer an einer Adresse etwas ändert, muss die Änderung ja bei den anderen Benutzern auch sichtbar sein. Da sie aber eine Kopie der Daten virtuell bei sich haben, müssen auch diese Daten aktualisiert werden.

Für die Realisierung dieses Problems wurde in der Beispieldatenbank eine Tabelle Zwischenspeicher erstellt, welche für diese Aktion verwendet wird, sowie eine globale Variable g_strChangeWatcher im Programm. Jedes Mal, wenn eine Adresse abgespeichert wurde, wird vom Programm auch der Zwischenspeicher-Wert mit einer GUID befüllt.

Das erste Mal beim Neustart der Applikation beinhaltet die g_strChangeWatcher einen Leerstring und stimmt mit dem Zwischenspeicher-Wert nicht überein. Dadurch wird ein erstes Requery gemacht und zudem die Variable mit dem aktuellen Zwischenspeicher-Wert befüllt.

Wird z.B. über ein Timer-Element die Refresh-Methode aufgerufen, vergleicht Ihr Programm die Variable mit dem Zwischenspeicher Wert. Istder Wert gleich, wurde nichts verändert und es erfolgt kein Refresh. Sobald jedoch eine Adresse geändert wird, wird auch der Zwischenspeicher-Wert neu gesetzt und alle anderen Benutzer erhalten im nächsten Prüf-Intervall die virtuelle Tabelle ge-refresht, da der Wert im Zwischenspeicher nicht mehr mit der g_strChangeWatcher übereinstimmt.

Beispiel für dieses Konzept:

```
Public Sub Requery()
Dim dlZS As DataLayer
Set dlZS = GetDataLayer() 'globale Funktion siehe Seite 4
dlZS.GetRecordset "SELECT * from Zwischenspeicher Where NameFeld = "
                  & Chr$(34) & "AdressChange" & Chr$(34)
If g_strChangeWatcher <> dlZS.FieldValue("WertText") Then
    dlAdressen.Refresh True
    lstAddressSearch.Clear
    dlAdressen.FillCBO lstAddressSearch, "Name", "ID", "Vorname",
                                          "Ort", False, True
     g_strChangeWatcher = dlZS.FieldValue("WertText")
End If
Set dlZS = Nothing
End Sub
```

Die Befüllung des Zwischenspeichers mit einer Kennung (z.B. GUID) kann beim Speichern vorgenommen werden.

7. Visualisierung von Daten

- FillCBO: Sie können dieser Methode ein Combobox oder ein Listbox-Objekt übergeben und mitteilen, mit welchen Daten diese Liste gefüllt werden soll. Grundsätzlich können Sie 3 Feld-Einträge pro Zeile anzeigen lassen. Prominentes Beispiel: Vorname Name, Ort. Dafür können Sie auch ein Komma an der richtigen Stelle setzen lassen. Ganz wichtig ist die Übergabe des Feldnamens, welches die ID beinhaltet (ist im Normalfall "id"). Diese ID wird nach vielen Leerzeichen angehängt, sodass sie in der Liste nicht sichtbar ist. Dies dient Ihnen jedoch dafür, dass Sie die ID des gewählten Listeneintrags ermitteln können.
  Beispiel: `lngID = Val(Trim(Right$(cboName,8)))`

- SetCBOPos: Diese Methode setzt in der übergebenen List- oder Combobox Objekt den ListIndex-Wert analog der aktuellen Zeilen-Position in der virtuellen Tabelle. Diese Funktion eignet sich sehr gut, um ein Listbox oder Combobox-Objekt mit dem RedMill DataBar Element zu steuern, mittels dem Event-Handler die aktuelle Position ausgelesen und im List/Combobox Element gesetzt werden kann.

- VisualizeToTextFile:
  Diese Methode zeigt Ihnen ein Editor-Fenster mit dem aktuellen Inhalt der virtuellen Tabelle des Datalayers.

- VisualizeToExcel:
  Analog VisualizeToTextFile wird eine Excel-Tabelle beschrieben und angezeigt. Wenn Sie die Option HandOverObject = True setzen, wird das Excel-Objekt zurückgegeben und die Tabelle bleibt unsichtbar. Dadurch können Sie weitere Formatierungen vornehmen und die Excel-Tabelle später selber anzeigen lassen. Auf diese Art und Weise können Sie sehr schnell Excel-Reports erstellen.

- WriteDataToCSV:
  Diese Methode ist der Kern der VisualizeToExcel-Methode und schreibt die Daten in ein CSV-File und gibt den Dateinamen zurück. Sie können anschliessend das CSV-File selber weiterbearbeiten.

- Excel: Mit dieser Eigenschaft wird ein neues ExcelTools-Objekt erzeugt oder das bestehende abgerufen. Ist kein Workbook geöffnet, wird automatisch die WriteDataToCSV Methode aufgerufen und das Workbook geöffnet. Sie können anschliessend im ExcelTools-Objekt mit dem Workbook weiterbearbeiten.

8. Schnittstelle zum clsOfflineRecordset

- GetOfflineRecordset:
  Über diese Eigenschaft können Sie das Offlinerecordset auskopplen. Beispiel:
  ```
  Set objORS = objDL.GetOfflineRecordset()
  ```
  oder Sie können direkt auf die Daten Zugreifen. Beispiel:
  ```
  lngRec = objDL.GetOfflineRecordset.RecordCount
  ```

- AttachOfflineRecordset:
  Diese Methode kann verwendet werden, wenn ein OfflineRecordset z.B. vom RedMill XML Handler zurückgegeben wird und für die weitere Arbeit im DataLayer verwendet werden soll. Sie verwenden diese Funktion auf eigene Gefahr, da das Offlinerecordset, welches attached wird, nicht der Kontrolle des DataLayers unterlag und daher nicht erforschte Probleme auftreten können. Wenn Sie ein OfflineRecordset von der XML-Komponente erhalten haben, können Sie dieses jedoch problemlos attachen. Wichtig ist, dass das OfflineRecordset auf ReadOnly gesetzt wird, d.h. es können keine Daten gespeichert werden.

  Damit ein OfflineRecordset attached werden kann, muss im OfflineRecordset-Objekt die Tag-Eigenschaft mit „AllowAttach“ gesetzt werden.

Klasse clsOfflineRecordset – Eigenschaften und Methoden

Sie können die clsOfflineRecordset Klasse selber instanzieren um mit einer virtuellen Tabelle zu arbeiten. Die virtuelle Tabelle hat als Index immer den Startwert 1 (nicht 0!).

1. Eigene Virtuelle Tabelle erzeugen und verwerfen

- Add: Die Add-Methode erzeugt die Angegebene Anzahl an clsOfflineRecord Objekten und speichert sie in einer Collection.
- FieldDim: Mit der FieldDim Methode werden in sämtlichen clsOfflineRecord Objekten die angegebene Anzahl clsOfflineField Objekte erzeugt und in der Collection der jeweiligen clsRecord Objekten gespeichert.
- Remove: Löscht das angegebene clsOfflineRecord Objekt.
- Destroy: Löscht die ganze virtuelle Tabelle inkl. allen Daten.

2. Zugriff auf die Records

- RecordCount: ReadOnly. Gibt die Anzahl der Zeilen in der virtuellen Tabelle zurück

- SetRecordCount:
  Setzt die RecordCount-Eigenschaft, wenn diese aus irgend einem Grund übersteuert werden soll.

- Records(Index): Gibt das gemäss Index angegebene clsOfflineRecord Objekt zurück. Möglichkeit zur Auskoppelung oder zum direkten Zugriff (Beispiel: `.Records(1).Fields(2).Value`)

- RecordCollection:
  Gibt das ganze Collection Objekt mit allen Records zurück.

- OverwriteRecordCollection:
  Mit dieser Methode kann ein Record-Collection Objekt angegeben werden, welches dann anstelle des bisherigen verwendet werden soll. Diese Methode wird von der Zoom Methode aus dem Extract-Objekt aufgerufen. Verwenden Sie diese Methode mit Vorsicht.

3. Arbeit mit einem DAO Recordset

- FillFromDAORecordset:
  Dieser Methode kann ein DAO Recordset übergeben werden. Damit erzeugt diese Methode selbstständig eine virtuelle Tabelle und befüllt sie mit den Daten aus dem DAO-Recordset. Diese Methode wird auch von der DataLayer Klasse verwendet und ist quasi der Kernel der Methode GetRecordset der DataLayer-Klasse. Wenn Sie die bolWithClose Eigenschaft nicht setzen, bleibt das Recordset offen, ansonsten wird das DAO-Recordset geschlossen.

- RecordsetIsReadOnly:
  Die Eigenschaft RecordsetIsReadOnly erhält den Status True, wenn das DAO-Recordset true ist. Diese Eigenschaft kann jedoch auch beschrieben werden, um dem DataLayer Objekt ein readonly Recordset vorzugaukeln.

4. Sortierfunktionen

- RecordsetSort: Diese Methode ist quasi der Kernel zur gleichnamigen Methode in der DataLayer-Klasse und ist dort beschrieben. Im Unterschied wird im clsOfflineRecordset nicht mit den FeldNamen sondern mit dem FeldIndex gearbeitet und daher muss auch der entsprechende Index übergeben werden.

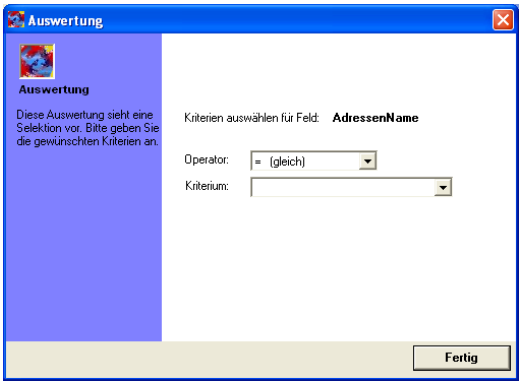
- RecordsetInnerSort:
  Diese Methode kann für nachfolgende Sortierwünsche verwendet werden, d.h. wenn innerhalb einer bestehenden Sortierung sortiert werden soll. Wichtig ist die Übergabe des zuletzt sortierten Felds, damit innerhalb dieses angegebenen Felds sortiert werden kann.

- DeleteOutfilteredRecords:

  Dies ist quasi der Kernel der Filter-Funktion in der DataLayer Klasse und kann selbstständig benutzt werden, mit dem Unterschied, dass der Spalten-Index übergeben werden muss und nicht der Feldname). Mit dieser Methode werden die Datensätze gelöscht, welche dem Filter entsprechen. Ein Backup der Daten wird hier nicht gemacht (dies wird von der entsprechenden DataLayer-Methode jedoch erstellt, wenn darüber verfahren wird).

Beispiel für die Verwendung des clsOfflineRecordset

Die clsOfflineRecordset Klasse kann dazu verwendet werden um beliebige Werte zu transportieren. Wenn immer Sie Werte in Ihrer Applikation speichern müssen, die in einer Tabelle gespeichert werden könnten, ist das clsOffineRecorset eine sehr gute Alternative.



In der separat erhältlichen RedMill Assistant Designer Komponente, mit welcher man Assistenten definieren und ausführen lassen kann, werden die Benutzereingaben in einem clsOfflineRecordset Objekt abgespeichert. Die jeweilige Applikation kann die Daten damit auslesen.

Klasse clsOfflineRecord – Eigenschaften und Methoden

Add:	Mit der Add-Methode kann die angegebene Anzahl Felder im aktuellen clsOfflineRecord Objekt erzeugt werden.
Remove:	Diese Methode löscht ein Feld innerhalb des aktuellen clsOfflineRecord Objekts.
FieldCount:	Gibt die Anzahl der Felder in dem aktuellen clsOfflineRecord Objekt zurück.
HasChanged:	Diese Eigenschaft kann verwendet werden um festzuhalten, ob ein Wert in dem aktuellen clsOfflineRecord geändert wurde.
NewlyAdded:	Diese Eigenschaft kann verwendet werden um festzuhalten, ob das aktuelle clsOfflineRecord Objekt neu hinzugekommen ist (dient zur separaten Behandlung des neu hinzugefügten Datensatzes).
Fields(Index):	Diese Eigenschaft gibt das gemäss Index erwünschte clsOfflineField Objekt zurück. Kann ausgekoppelt werden oder direkt verwendet werden (Beispiel: `.Fields(1).Value`)

- MoveFieldToNewPos:
  Diese Methode erlaubt es, ein Feld an eine andere Position zu bringen. Beispiel: aus "ID,Name,Vorname" wird "ID,Vorname,Name". Gilt nur für den aktuellen Record. Angabe der aktuellen Feldposition und der neuen Feldposition wird benötigt. Zu beachten ist, dass zuerst das Feld aus der Collection entfernt und anschliessend neu hinzugefügt wird.

  Der Vorteil dieser Methode ist, dass horizontal sortiert werden kann. Die Konsistenz der virtuellen Tabelle muss allerdings von Ihnen selber gewährleistet werden.

Klasse clsOfflineField – Eigenschaften und Methoden

- Value: Variant-Eigenschaft, kann irgend einen Feldwert beinhalten
- FieldStatus: Schreib-Eigenschaft, mittels welcher ein Boolean-Wert gesetzt wird.
- Status: Lese-Eigenschaft der gesetzten FieldStatus Eigenschaft
- Anything: Diese Eigenschaft gibt die Value-Eigenschaft zurück, sofern der FieldStatus nicht auf True gesetzt wurde, da bei FieldStatus=True der Status-Wert zurückgegeben wird.
- Description: Beschreibung oder Feldname (wird von der DataLayer Klasse benutzt um die Feldnamen zu hinterlegen).
- HasChanged: Boolean-Eigenschaft, welche dazu verwendet werden kann um festzuhalten, ob in dem Feld etwas geändert wurde.
- lngFieldType: Mit dieser Eigenschaft kann der Feldtyp festgelegt und abgefragt werden.
- IsStatusSet: Mit dieser Lese-Eigenschaft kann abgefragt werden, ob der Status verwendet wurde

Arbeiten mit Extrakten

Unter Punkt 4 des DataLayers ist eine Filtermethode beschrieben, welche einen entscheidenden Nachteil hat. Die Daten im Filter müssen speziell gesichert werden, da sie sonst nicht in die Datenbank geschrieben werden. Nicht so bei der Extract-Möglichkeit. Hier findet eine Verlinkung statt, sodass die bearbeiteten Daten im Extract direkt mit dem Datalayer verbunden sind. Eine Änderung der Daten im Extract erfolgt direkt auch im clsOfflineRecordset Objekt, da im Extrakt nur eine Referenz zum entsprechenden Record gesetzt wird. Für den Extract (oder für mehrere Extracts) stehen zwei Klassen zur Verfügung: clsExtractManager und clsExtract. Beide Klassen sollten nicht eigenständig instanziert werden sondern über die im DataLayer verfügbare Methode.

Klasse clsExtractManager – Eigenschaften und Methoden

- ExtractRecords:
  Bei dieser Methode geben Sie die gewünschte Filtereinstellung an und erhalten einen Objekt-Handle vom Typ Long. Diesen ObjectHandle sollten Sie in einer Variable ablegen, denn damit haben Sie Zugriff auf den Extrakt.

- Extracts: Diese Eigenschaft überigbt ein clsExtract Objekt gemäss dem übergebenen Objekt-Handle.

- DestroyExtract:
  Das clsExtract Objekt wird auf Nothing gesetzt, der Objekt-Handle existiert nicht mehr.

- ParentDL: Write-Only Eigenschaft für interne Verwendung.

Klasse clsExtract – Eigenschaften und Methoden

Die clsExtract Klasse beinhaltet vor allem Funktionen um mit den extrahierten Daten zu arbeiten. Die folgenden Eigenschaften und Methoden sind identisch wie diejenige im DataLayer und unter diesem Kapitel beschrieben.

- FindRecord
- MoveFirst
- MoveLast
- MoveNext
- MovePrevious
- MoveSpecific
- RecordCount
- VisualizeToTextFile
- FillCBO

- ObjectHandle Diese Eigenschaft setzt oder gibt den Objekt-Handle zurück.

- SetExtract Mit dieser Write-Only Eigenschaft kann ein Collection-Objekt mit den clsOfflineRecord-Objekten übergeben werden. Interne Verwendung.
- Analyzer Gibt ein Analyzer-Objekt zurück

- SubductRecords: Mit dieser Methode können Datensätze aus dem Extrakt entfernt werden. Mit der Übergabe des Feldnamens, des Operators und des Werts werden alle Datensätze die zutreffen aus dem Extract entfernt.

- Zoom: Diese Methode erlaubt, den aktuellen Extrakt auf den Datalayer zu übernehmen. Sämtliche Extrakte werden dabei gelöscht. Mit dieser Methode können Sie den Extrakt normal im Datalayer weiterverarbeiten. Bitte geben Sie an, dass Sie wissen, was Sie tun (I know what I do) = True

Die clsAnalyzer-Klasse

Mit der Analyzer-Klasse, welche Sie vom DataLayer oder von einem Extrakt aus aufrufen können, werden Zusammenzüge über die Daten gemacht. Zusätzlich zur CalcSum Methode im DataLayer können Sie mit der Analyzer-Klasse folgende Datenanalysen machen:

- OccurrenceCount:
  Gibt die Anzahl des Auftretens eines Werts innerhalb des Offlinerecordsets zurück. Übergabe der Parameter FieldName und SearchValue. FieldName steht für den Feldnamen (Spalten-Namen in dem gesucht wird). SearchValue= Wert, nach dem gesucht wird. Die Rückgabe ist ein Wert vom Typ Long, welcher die Anzahl zurückgibt, wievielmal der Wert in der Spalte vorkommt. Ohne MatchCase wird die Gross/Kleinschreibung bei der Suche ignoriert.

- SingleOccurrence:
  Erstellt eine Liste aller im OfflineRecordset enthaltenen Werte von übergebenen Spalte und löscht alle doppelt vorkommenden Werte. Der Rückgabewert ist ein Collection Objekt.

- ReplaceField: Sucht die SourceValue und ersetzt die Werte mit TargetValue innerhalb einer Spalte gemäss FieldName. Mit StringCompare kann der Vergleich nur mit Text erfolgen (Anfangs- und Ende-Leerzeichen werden ignoriert). Ohne MatchCase wird die Gross/Kleinschreibung bei der Suche ignoriert.

- Pivot: Mit der Pivot-Funktion kann ein Zusammenzug der Daten nach einem Kriterium erfolgen. Die Felder werden in einem String mit Semikolon abgetrennt übergeben. Alle übergebenen Felder werden auch ausgegeben. Das CriteriaField bestimmt, nach welchem Feld der Zusammenzug erfolgen soll. Die Funktion erstellt eine Summe pro Wert, der in der Kriterienspalte vorkommt und zwar für alle übergebenen Spalten. Spalten mit String-Werten werden bei der Ausgabe eine 0 erhalten. Beispiel: eine Liste mit Umsätzen pro Kunde. Nun soll der Umsatz pro Land ausgegeben werden. Dann kann als ValueFieldsDividedBySemicolon = „Land;Umsatz“ angegeben werden, wenn diese Spalten so benannt sind. Das CriteriaField ist dann „Land“.
  Der Rückgabewert ist ein DataLayer-Objekt mit einer virtuellen Tabelle. Diese Tabelle kann dann z.B. direkt in ein Excel ausgegeben werden.

- CalculateFields: Mit dieser Methode können Sie ein Feld und ein anderes zusammenrechnen und in ein drittes Feld hineinschreiben. Die Operation wird über alle Datensätze vorgenommen. Nehmen wir an, Sie haben ein Feld „Umsätze“, ein Feld „UmsäteVorjahr“ und ein Feld „Vergleich“. Sie haben jedoch nur die Daten in den ersten beiden Feldern drin. Mit dieser Funktion können Sie für jeden Datensatz z.B. Umsätze – UmsätzeVorjahr rechnen und so den Vergleich in das Feld „Vergleich“ setzen lassen.

- CalculateFieldsAndValue1: Gleich wie CalculateFields, einfach mit dem Unterschied, dass Sie ein Feldname und einen fixen Wert rechnen können. Z.B. Feldname - 20

- CalculateFieldsAndValue2: Gleich wie CalculateFieldsAndValue1, einfach umgekehrt Z.B. 20 - Feldname

- CalculateFormula: Mit dieser Funktion können Sie einen beliebigen String übergeben mit einer Rechen-Anweisung. Dabei können Feldnamen in <Klammern> (Grösser/Kleiner-Zeichen) übergeben werden. Es sind die gängigen Operatoren erlaubt wie + - * / ^ u.s.w.

  Beispiel: Sie haben eine Tabelle mit dem Feld Umsatz, UmatzVorjahr und dem Feld Differenz in Prozent. Sie können jetzt z.B. folgenden Aufruf machen:

```
CalculateFormula "1-<Umsatz>/<UmsatzVorjahr>", "Differenz", 2
```

  Der Schalter „WantHelp“ gibt Ihnen einen Hilfetext aus, der ein ähnliches Beispiel zeigt.

Die Formula-Klasse

Die CalculateFormula Methode ruft die Funktionalität in der Formula Klasse auf. Diese Klasse kann selbständig instanziert werden und kann einen Formeltext parsen und auswerten. Man übergibt zuerst den Formeltext (FormulaText) und ruft dann die Funktion auf, welche dann einen Wert vom Typ Double zurückgibt. Es ist hier eine Rundung implementiert, d.h. man könnte z.B. mit Rnd2(0.0025*1) den Wert auf 2 Stellen runden.

Klasse clsTableManager – Eigenschaften und Methoden

Die Klasse clsTableManager dient zur einfachen Handhabung der verschiedenen DAO-Funktionen und kann selbstständig instanziert werden.

- CreateDatabase: Erzeugt eine neue MS Access-Datenbank mit dem angegebenen Pfad. Die Erzeugung einer Datenbank mit Workgroup wird nicht unterstützt.
- CreateTable: Erzeugt eine neue Tabelle in der Datenbank mit dem angegebenen Namen.
- AppendField: Erzeugt das angegebene Feld in der angegebenen Datenbank mit dem angegebenen Feldtyp.
- SetIndex: Erzeugt einen Index in der Datenbank.
- DeleteTable: Löscht die angegebene Tabelle in der Datenbank.
- CreateQueryDef: Erzeugt ein neues Query in der Datenbank mit dem angegebenen Namen.
- DeleteQueryDef: Löscht das angegebene Query in der Datenbank.
- GetFieldType: Holt den Feldtyp des angegebenen Felds aus der angegebenen Tabelle und gibt diesen zurück.
- ListQueryFields: Diese Methode listet alle Feldnamen aus einem Query auf und schreibt sie in eine Collection. Gibt die Collection zurück.
- ErrorStatus: Allfällige Fehler können über den Errorstatus abgefragt werden.
- DBUpdate: Erstellt ein clsUpdateDB Objekt und gibt dieses zurück

Klasse clsUpdateDB – Eigenschaften und Methoden

1. Allgemein

Wenn neue Funktionen in eine Software implementiert werden, müssen z.B. zusätzliche Felder in der Datenbank vorhanden sein, damit die neuen Funktionen überhaupt funktionieren. Das Anlegen oder Ändern von Tabellen und Feldern in der Datenbank kann vollautomatisch durchgeführt werden, dies anhand einer Versionskontrolle.

Die clsUpdateDB Klasse kümmert sich genau darum, d.h. man übergibt eine Liste von Aktionen, die durchgeführt werden sollen. Mit der Zeit nimmt wird die Liste immer länger, da immer mehr Aktionen durchgeführt werden müssen. Man weiss ja nicht, welcher Status die Datenbank bei den verschiedenen Benutzern hat. Daher ist eine Versionskontrolle der Datenbank erforderlich. Hat die Datenbank bei einem Benutzer eine bestimmte Version, dürfen die früheren Datenbank-Updates natürlich nicht mehr ausgeführt werden.

Die clsUpdate notiert sich die ausgeführten Aktionen. Wenn nun eine neue Version Ihrer Software neue Datenbank-Updates erfordern, weiss diese Klasse bereits, ab welcher Aktions-Nummer die Datenbank-Updates für die jeweils spezifische Kundendatenbank ausgeführt werden müssen.

Das Ziel ist, dass Sie immer sämtliche Datenbank-Updates auch von alten Versionen übergeben, da bei den Benutzern draussen ev. noch eine Datenbank vorkommt, welche auf einem Uralt-Stand ist und daher müssen dort auch sämtliche Updates durchgeführt werden.

Auf diese Art und Weise können Sie es sich sparen, eine Access-Datenbank mitzuliefern, da Ihr Programm über den DataLayer eine neue Datenbank erzeugen und mit der clsUpdateDB die Struktur erstellen kann.

2. Eigenschaften und Methoden

Parent: Beinhaltet das clsTableManager Objekt (muss zwingend vorhanden sein, daher sollte diese Klasse auch nicht selbstständig instanziert werden sondern immer über die Eigenschaft vom clsTableManager aufgerufen werden).

- AppendUpdateAction:

  Mit dieser Methode können Sie die auszuführenden Aktionen übergeben. Sie müssen in Ihrer Funktion der Datenbank-Updates also nur noch die Updates deklarieren, der Rest übernimmt diese Klasse. Code-Beispiele:

```
With objTableManager.DBUpdate
    .AppendUpdateAction 1, 1, "MyTable"
    .AppendUpdateAction 2, 3, "MyTable", "ID", 32
    .AppendUpdateAction 3, 3, "MyTable", "Field2", 10
End With
```

  Die Update-Anweisungen müssen zwingend fortlaufend nummeriert werden (erste Zahl). Die Zweite Zahl ist die Aktion, welche ausgeführt werden soll:

```
dbupdate_CreateTable = 1
dbupdate_DeleteTable = 2
dbupdate_AppendField = 3
dbupdate_DeleteField = 4
dbupdate_SetIndex = 5
```

  Danach folgen 3 Optionen: Die erste Option ist der Tabellen-Name und muss zwingen angegeben werden. Die 2. Option ist der Feldname und die 3. Option ist entweder der Felddatentyp oder bei SetIndex ein Boolean-Wert ob Primärschlüssel oder nicht.
  Im Übrigen werden die Aktionen über den clsTableManager gesteuert. Man kann bei diesen Methoden abschauen, welche Parameter übergeben werden müssen.

  DB-Feldtypen:

```
dbtypeBoolean = 1
dbtypeDouble = 7
dbtypeLong = 3
dbTypeLongAutoValue = 32
dbtypeText = 10
dbtypeMemo = 12
dbtypeDate = 8
```

- UpdateDatabase: Mit dieser Methode wird das Datenbank-Update gestartet und die mit der obigen Methode übergebenen Aktionen abgearbeitet. Ist in der Datenbank die Tabelle `sys_redmilldatalayer` nicht vorhanden, wird sie angelegt. Diese Tabelle dient für die Versionskontrolle.
  Allfällige Fehler werden direkt angezeigt und das Datenbank-Update abgebrochen.

Klasse Excel-Tools – Eigenschaften und Methoden

Diese Wrapper-Klasse um ein Microsoft Excel Workbook herum kann separat oder in Zusammenhang mit dem DataLayer verwendet werden (siehe Excel-Methode).

- WBOpen: Öffnet eine bestehende Microsoft Excel Arbeitsmappe
- WBNew: Erstellt eine neue leere Microsoft Excel Arbeitsmappe
- WBClose: Schliesst eine geöffnete Excel-Arbeitsmappe. Das Excel-Objekt kann dabei offen gehalten werden, wenn im gleichen ExcelTools-Objekt eine andere Arbeitsmappe geöffnet werden soll. Ansonsten wird das Excel-Objekt ebenfalls geschlossen
- WBObject: Gibt ein Workbook-Objekt zurück
- IsWorkbookOpen: Diese Eigenschaft ist True, wenn eine Arbeitsmappe offen ist
- InjectMacroByFile: Lädt eine Text-Datei mit einem Makro drin und erstellt im aktuell geöffneten Workbook-Objekt ein neues Code-Modul mit dem Makro aus dem File. Ein Makro muss immer Sub Name() und End Sub beinhalten, damit es ausgeführt wird. Der Name kann frei gewählt werden.
- InjectMacroByString: Analog wie die obige Makro-Funktion wird ein neues Code-Modul mit dem in einem String übergebenen Makro erzeugt. Beachten Sie, dass Sie selber die CrLf im String mitgeben müssen, wenn im String mehrere Zeilen vorhanden sind.
- ShowApp: Zeit das Excel-Workbook an oder blendet es wieder aus (Excel nicht mehr sichtbar). WriteOnly-Eigenschaft
- ClearErr: Löscht den ErrorStatus
- ErrorStatus: Allfällige Fehler können über den Errorstatus abgefragt werden.
- Names: Erstellt ein neues clsNames-Objekt und erlaubt Ihnen, auf diese Methoden und Eigenschaften zuzugreifen.

Klasse clsNames – Eigenschaften und Methoden

Mit dieser Klasse können Sie auf die Bereichs-Namen in Microsoft Excel zugreifen. Die Klasse kann nicht selbstständig instanziert werden, da sie auf das Workbook-Objekt in der ExcelTools Klasse zugreift. Daher stellt die ExcelTools Klasse eine Eigenschaft zur Verfügung, damit Sie auf das clsNames Objekt zugreifen können.

Count:	Gibt die Anzahl der definierten Namen im Workbook-Objekt zurück
List:	Gibt ein Collection-Objekt zurück mit der Liste aller definierten Namen im Workbook-Objekt
SourceRange:	Gibt den Zell-Range zurück für den der definierte Name gilt
ValueOfName:	Gibt den Zellwert zurück für den der definierte Name gilt. Achtung: Beinhaltet ein Name mehrere Zellen, kann kein Wert zurückgegeben werden. Es entsteht zwar kein Fehler aber Sie erhalten keinen Rückgabewert.
Parent:	Zugriff auf das ExcelTools Objekt, von dem aus das aktuelle clsNames Objekt erstellt wurde.

Testprogramm

Das mitgelieferte Testprogramm gibt Ihnen einen Einblick in die Handhabung der Methoden und Eigenschaften des Datalayers. Es handelt sich um VB-Projekt, welches Sie sehr einfach in VisualBasic 6 laden können. Sollten Sie nicht über Visual Basic verfügen, wird empfohlen, das EXE-Programm auszuführen und die frmTest Datei über ein Editor-Fenster zu öffnen. Auf diese Art und Weise können Sie die Arbeitsweise sehr gut nachvollziehen.

Das RedMill DatalayerControlBar Steuerelement

Dieses Steuerelement wird nicht mehr weiterentwickelt. Verwenden Sie bitte das FormDataControl wie weiter unten beschrieben

Dieses Steuerelement wurde dem DAO Data Steuerelement nachempfunden und macht im Grundsatz das gleiche. Zudem wurde direkt eine (einschaltbare) Suche integriert, wenn der Benutzer auf das Element im weissen Bereich klickt. Der Databar arbeitet mit dem Startwert 1 und dem maximalen-Wert, der über eine Eigenschaft gesetzt werden kann. Die unterstützte Sprache ist Deutsch. Dieses Steuerelement verfügt über eine Help Methode, welche einen Hilfetext zurückgibt, welcher insbesondere dann von Nutzen ist, wenn Sie mit dem Element ohne den DataLayer arbeiten möchten. Diese Dokumentation beschränkt sich auf den Teil der Arbeit mit dem DataLayer.

Die Eigenschaften, Events und Methoden des RedMill DataLayerControlBar sind wie folgt:

1. Aussehen

   Max: (ReadWrite) Setzt oder liest den Max-Wert des Elements. Wird in Zusammenhang mit dem DataLayer automatisch gesetzt

   DisplayText: Setzt den Text, der die Datenart angibt (in der oberen Abbildung wurde die Eigenschaft auf "Adressen" gesetzt, was im Element entsprechend angezeigt wird.

   ShowNameOrFieldName:
   Bestimmt, welcher Feldname ausgegeben wird (in der obigen Abbildung entspricht dies dem 'Diverse', denn es wurde eine Adresse mit Namen "Diverse" erfasst). Sie können auch mehrere Felder mit Semikolon getrennt übergeben (Beispiel: `dlcAdressen.ShowNameOrFieldName = "Name;Vorname"`)
   Wichtig zu wissen ist, dass der erste Feldname für die Filterung verwendet wird!

   SearchActivated:
   Wenn auf True gesetzt, wird die Suche angeboten, ansonsten wird die Suche nicht angeboten. Die Suche wird dann ausgelöst, wenn der Benutzer in den weissen Bereich klickt und einen Suchwert eingibt und mit Enter bestätigt. Eigentlich wird jedoch keine Suche ausgelöst sondern im DataLayer die Filter-Methode aufgerufen. Das Databar Element zeigt dann nur noch die Anzahl Datensätze, welche gefiltert wurde. Zudem wird die Schaltfläche angezeigt, mittels welchem der gesetzte Filter gelöscht werden kann.

   HideDeleteFilterButton:
   Verhindert die Anzeige der Schaltfläche, mit welcher ein gesetzter Filter wieder gelöscht werden kann.

   ShrinkWidth: Mit der ShrinkWidth Methode kann die Breite des Elements gesteuert werden, d.h. das Element auf eine bestimmte Breite gesetzt werden. Da die Methode ShrinkWidth (verkleinern)

heisst, muss natürlich bei einer gewünschten Verbreiterung ein Minus-Wert angegeben werden (z.B. Verschmälerung um –1500 entspricht einer tatsächlichen Verbreiterung)

Position: Diese ReadWrite-Eigenschaft setzt oder gibt die aktuelle Position zurück. Mit dem Beschreiben der Eigenschaft wird der Cursor im DataLayer auf die entsprechende Zeile gesetzt.

2. Verbindung mit dem RedMill DataLayer

DataLayerBind: Dieser Eigenschaft können (resp. müssen) Sie das DataLayer Objekt übergeben, damit das DataBar Steuerelement den DataLayer steuern kann.

RefreshBinding:
Diese Methode muss aufgerufen werden, wenn z.B. im DataLayer ein Datensatz hinzugefügt oder gelöscht wurde. In diesem Fall wird der Max-Wert automatisch wieder von der RecordCount Eigenschaft im DataLayer gesetzt sowie das DataBar Element auf den 1. Datensatz gesetzt.

3. Steuerung von Textfeldern

AppendTextBox: Diese Methode erlaubt, beliebige Textbox-Objekte (oder Objekte, welche die Methode .Text und .DataChanged aufweisen) am DataBar „anzuhängen". Gleichzeitig muss der Feldname angegeben werden. Das Verhalten ist ähnlich wie die DataSource Eigenschaft des Textbox-Elements selber, d.h. der DataLayerBar wird automatisch die Anzeige der Daten im entsprechenden Textfeld und die automatische Speicherung von geänderten Daten im Datalayer vornehmen. Auf diese Art und Weise erstellen Sie sehr rasch eine Datenbank-Applikation.

4. Ereignisse

RedMillDatabarBeforeChange:
Dieses Ereignis wird dann ausgelöst, wenn der Benutzer auf eine Weiter oder Zurück-Schaltfläche klickt, bevor jedoch die aktuelle Position neu gesetzt wird. Auf diese Art und Weise können Sie z.B. Daten in Textfeldern in den DataLayer auf der bestehenden Position abspeichern.

RedMillDatabarChange:
Dieses Ereignis wird nach dem vorigen Ereignis ausgelöst, allerdings wurde hier die Position bereits neu gesetzt, was Ihnen erlaubt, neue Daten aus dem DataLayer herauszulesen und anzuzeigen.

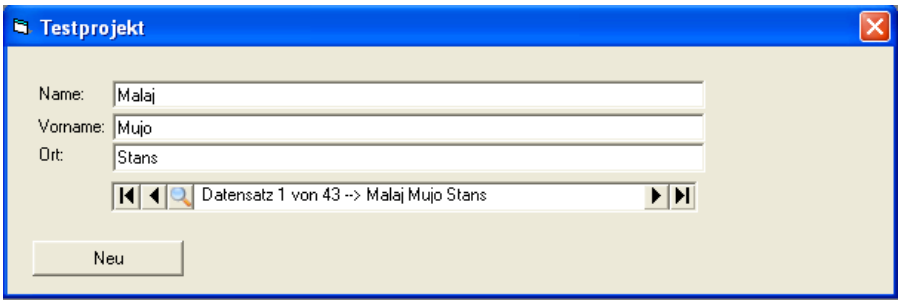
RedMillDatabarSearch:
Dieses Ereignis wird dann ausgelöst, wenn ein Benutzer eine Suche gestartet wird. Der Suchtext wird beim Ereignis übergeben und kann so für weitere Aktionen verwendet werden.

Das RedMill FormDataControl Steuerelement

Dieses Steuerelement ist im Prinzip der Nachfolger des DatalayerControlBar und bietet komfortablere Möglichkeiten. Zudem arbeitet es nicht mit der Filter-Methode sondern mit der Extrakt-Methode, was die Abspeicher-Problematik löst.

Mit diesem Steuerelement haben Sie im Handumdrehen eine Eingabemaske erstellt. Folgendes Testprojekt soll dies verdeutlichen.

Wir haben hier eine ganz einfache Adressverwaltung. Wir haben ein Formular mit 3 Feldern sowie dem RedMill FormDataControl.



Die Applikation läuft mit folgendem Code reibungslos:

Wir deklarieren die globale Variable

```
      Dim g_dlAdressen As DataLayer

Private Sub Form_Load()
      Set g_dlAdressen = GetDataLayer()    ´zentrale Instanzierungsfunktion
      g_dlAdressen.SetDatabaseByPath = "<meine Datenbank>"
      g_dlAdressen.GetRecordset "select * from adressen"
      ctlFormData.BindDataLayer g_dlAdressen, False, False, True
      ctlFormData.AppendTextBox txtName, "Name"
      ctlFormData.AppendTextBox txtVorname, "Vorname"
      ctlFormData.AppendTextBox txtOrt, "Ort"
      ctlFormData.SearchListConfig "Name", "Vorname", "Ort", True, False
      ctlFormData.ShowNameOrFieldName = "Name;Vorname;Ort"
      ctlFormData.ShrinkWidth 2500
End Sub
```

Mit den obigen Zeilen ist die ganze Applikation programmiert. Was noch fehlt ist eine Neu-Schaltfläche und am Schluss natürlich beim Beenden noch die Save-Methode wie folgt:

```
Private Sub Form_Terminate()
      ctlFormData.Position = 1
      g_dlAdressen.Save
      set g_dlAdressen = nothing
end sub
```

Das Steuerelement selber hat direkt eine Filter-Möglichkeit, wenn man auf den weissen Balken klickt oder eine Volltextsuche wenn man auf das Such-Symbol klickt.

Eigenschaften und Methoden

- AppendTextBox: Diese Methode erlaubt, beliebige Textbox-Objekte (oder Objekte, welche die Methode .Text und .DataChanged aufweisen) am Steuerelement „anzuhängen“. Gleichzeitig muss der Feldname im DataLayer angegeben werden. Das Verhalten ist ähnlich wie die DataSource Eigenschaft des Textbox-Elements selber, d.h. der DataLayerBar wird automatisch die Anzeige der Daten im entsprechenden Textfeld und die automatische Speicherung von geänderten Daten im Datalayer vornehmen. Auf diese Art und Weise erstellen Sie sehr rasch eine Datenbank-Applikation.

- BindDataLayer: Mit dieser Eigenschaft übergeben Sie das DataLayer Objekt. Zudem können Sie optional die folgenden Angaben übergeben:
  - bolSearchDisabled = Suche aktiviert oder inaktiv (zeigt das Such-Symbol an oder nicht). True = Disabled, d.h. wird nicht angezeigt. Der Standard ist mit Anzeige
  - bolFilterDisabled = Filter aktiviert oder inaktiv. Auch hier ist der Standardwert aktiv.
  - SuppressMessage: Beim Aufruf dieser Methode wird standardmässig eine Hilfe-Mitteilung ausgegeben. Diese ist nur für den Programmierer bestimmt, sodass Sie hier den Wert immer auf True setzen sollten.

- RefreshBinding: Wenn im DataLayer etwas geändert wurde (z.B. neuer Datensatz oder Refresh im DataLayer), können Sie auf diese Art und Weise das Steuerelement auch auf den neusten Stand bringen.

- bolSetTextToManual: Mit dieser Methode wird der Anzeigetext statisch angezeigt und nicht mehr dynamisch

- DisplayText: Wenn Sie die obere Eigenschaft verwenden, können Sie mit dieser Methode den Anzeigetext einstellen.

- Help: Gibt einen Hilfetext zurück

- ShrinkWidth: Lässt die Breite des Elements verkleinern oder mit einem Minuswert vergrössern

- HideDeleteFilterButton: Mit dieser Methode kann nachträglich die Filter-Löschen-Schaltfläche versteckt oder wieder angezeigt werden. So kann z.B. erreicht werden, dass eine Filterung nicht mehr widerrufen werden kann.

- SearchListConfig: Mit dieser Methode kann die Volltextsuche konfiguriert werden, d.h. welche Felder angezeigt werden sollen. Die Feldnamen werden als String mit Semikolon getrennt übergeben.

- FieldList: Mit dieser Eigenschaft kann die Feldliste aus SearchListConfig abgerufen werden.

ShowNameOrFieldName:
Dies ist die Konfiguration der Anzeige, d.h. welche Feldnamen im Steuerelement selber angezeigt werden sollen.

Max: Diese Eigenschaft brauchen Sie nur, wenn Sie ohne DataLayer arbeiten. Sie können die maximale Anzahl an Records setzen. Verwenden Sie diese Eigenschaft nie in Zusammenhang mit dem DataLayer

Position: Mit dieser Eigenschaft kann die Record-Position gesetzt werden. Dadurch wird im DataLayer ebenfalls die Position neu gesetzt. Mit dieser Eigenschaft werden auch alle Eingaben in den Textfeldern in den DataLayer automatisch abgespeichert.

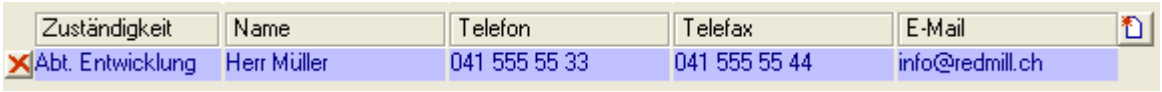
Ereignisse

RedMillDatabarBeforeChange:
Dieses Ereignis wird dann ausgelöst, wenn der Benutzer auf eine Weiter oder Zurück-Schaltfläche klickt, bevor jedoch die aktuelle Position neu gesetzt wird. Auf diese Art und Weise können Sie z.B. Daten in Textfeldern in den DataLayer auf der bestehenden Position abspeichern.

RedMillDatabarChange:
Dieses Ereignis wird nach dem vorigen Ereignis ausgelöst, allerdings wurde hier die Position bereits neu gesetzt, was Ihnen erlaubt, neue Daten aus dem DataLayer herauszulesen und anzuzeigen.

RedMillDatabarSearch:
Dieses Ereignis wird dann ausgelöst, wenn ein Benutzer eine Suche gestartet wird. Der Suchtext wird beim Ereignis übergeben und kann so für weitere Aktionen verwendet werden.

Das RedMill DataGrid Steuerelement



Das RedMill DataGrid Steuerelement hat 5 Spalten und 9 Zeilen. Grundsätzlich können beiliebige Zeilen abgebildet werden, es werden jedoch nie mehr als 9 Zeilen angezeigt. Die Anzeige von mehr als 5 Spalten ist nicht möglich. Vertikal ist ein Scrollbalken vorgesehen, jedoch horizontal nicht. Mit der DataBind Eigenschaft kann ein DataLayer Objekt übergeben werden, welches vom Grid gesteuert wird (neue Datensätze anlegen, Datensätze löschen, Daten ändern, u.s.w.).

1. Visuelle Steuerung

DataBinding: Dieser Eigenschaft können (resp. müssen) Sie das DataLayer Objekt übergeben, damit das DataGrid Steuerelement den DataLayer steuern kann.

Editable: Mit dieser Eigenschaft steuern Sie, ob der Benutzer im Steuerelement Daten verändern darf oder nicht. Mit Editable werden automatisch die Eigenschaften AllowNew und Deletable auf True gesetzt.

AllowNew: Die Schaltfläche Neu wird angezeigt oder nicht.

EnableNewRecord:
Es wird ein interner Code gesetzt, der ev. später von Bedeutung ist, momentan aber keine Bedeutung hat.

Deletable: Zeigt die Lösch-Schaltflächen an oder nicht.

HideShowCol: Zeigt die angegebene Spalte gemäss Index an oder nicht, je nachdem wie der bolInvisible Wert gesetzt wird (True = nicht anzeigen!).

ColWidth: Diese Methode setzt bei der angegebenen Spalte gemäss Index den angegebenen Width-Wert (Spaltenbreite)

ScrollbarLeft: Der Scrollbalken ist auf die normale Breite ausgelegt. Wenn nun Spalten auf unsichtbar gesetzt werden oder das Element verbreitert wird, muss der Scrollbalken (und damit auch die Neu-Schaltfläche) verschoben werden. Mit der lngLeft Angabe können Sie den Left-Wert des Scrollbalkens angeben. Sie müssen etwas ausprobieren, um den genauen Wert zu ermitteln.

ShowMaxRows: Legt die Anzahl sichtbarer Zeilen fest (max. 9). Wenn z.B. nur 3 Zeilen angezeigt werden sollen, können Sie mit dieser Eigenschaft dies bewerkstelligen.

- VisualizeDataLayer:
  Dies ist die Haupt-Methode des DataGrid Elements. Damit lässt sich die virtuelle Tabelle des Datalayer in einer Anweisung anzeigen. Minimal muss ein Feldname übergeben werden, maximal können 5 Feldnamen übergeben werden. Zudem kann der Spalten-Index einer Spalte übergeben werden, welche die booleischen Werte in Ja/Nein Werte konvertiert haben soll sowie zwei Spalten, welche die Zahlen mit dem Format #'##0.00 formatiert bekommen sollen. Wenn bolNoHeader auf True gesetzt wird, werden die Spaltenbeschriftungen nicht neu aufbereitet. Auf diese Art und Weise kann auch ein Refresh vorgenommen werden, falls Sie im DataLayer Daten geändert haben.

- Clear: Die Clear-Methode löscht alle Zelleninhalte, tangiert jedoch das übergebene DataLayer Objekt nicht.

- HeaderDescription:
  Die Visualize DataLayer-Methode setzt auch die Spaltenüberschriften gemäss den in der Abfrage vorhandenen Feldnamen. Mit HeaderDescription kann im angegebenen SpaltenIndex die Spaltenbeschriftung gesetzt werden.

- AlignRight: Setzt die angegebene Spalte auf rechtsbündig.

- ConvertNumberFormat:
  Dies ist eine interne Methode, welche von der Methode VisualizeDataLayer aufgerufen wird, welche Sie aber ev. Für andere Zwecke verwenden können, daher ist sie verfügbar. Die Methode gibt den formatierten Wert zurück und hat daher keinen Einfluss auf die Steuerung.

- CellContent: Die CellContent Eigenschaft gibt von der Zelle der angegebenen Koordinaten (SpaltenIndex und ZeilenIndex) den aktuellen Wert zurück.

- CurrentRow: Gibt den Zeilenindex der aktiven Zeile zurück, d.h. von der Zeile, wo der Benutzer den Cursor gesetzt hat.

- CurrentCol: Gibt den Zeilenindex der aktiven Spalte zurück, d.h. von der Spalte, wo der Benutzer den Cursor gesetzt hat.

2. Informationen zur Speicherung

Da das DataGrid bei einer editierbaren Tabelle die Speicherung der geänderten Daten im Datalayer selber vornimmt, muss dem RedMill DataGrid Element die für die notwendigen ForceFields (siehe entsprechende Methode im Kapitel DataLayer) übergeben werden.

- AddForceField: Setzt die ForceFields im DataLayer

- ForceSaveCurrentField:
  Die Speicheraktion wird beim Verlassen der Zelle, d.h. beim Cursor-Wechsel veranlasst. Wenn Sie das Formular mit dem DataGrid Element schliessen kann es sein, dass der Benutzer in der Zelle etwas geändert, jedoch keinen Cursor-Wechsel gemacht hat. Dafür können Sie diese Methode aufrufen, welche auf jeden Fall das aktuelle Feld noch abspeichert, sollte etwas geändert worden sein.

3. Ereignisse

- CellChange: Dieses Ereignis wird dann ausgelöst, wenn in einer Zelle vom Benutzer etwas geändert wurde.

- CellCursorChange:
  Dieses Ereignis wird dann ausgelöst, wenn der Cursor vom Benutzer auf eine andere Zelle gesetzt wurde. Über Row und Col können Sie die aktuellen Koordinaten der neuen Zelle ermitteln.

- CellDoubleClick: Dieses Ereignis wird dann ausgelöst, wenn der Benutzer auf eine Zelle doppelklickt. Dieses Ereignis können sie dazu nutzen, um z.B. in eine Detailmaske zu springen und dort weitere Daten bearbeiten zu lassen. Sie erhalten die Koordinaten der Zelle, auf welche doppelgeklickt wurde sowie den Zellinhalt.

- CellSaved: Nachdem ein Zellinhalt im DataLayer Objekt abgespeichert wurde, wird dieses Ereignis ausgelöst.

- BeforeDelete: Dieses Ereignis wird vor der Löschaktion ausgelöst und kann dafür verwendet werden, wenn vor der Löschung abgefragt werden soll, ob der Benutzer wirklich löschen möchte. Wenn Sie `Cancel=True` setzen, wird die Löschaktion nicht durchgeführt. Über den Row-Index können Sie ermitteln, welche Zeile gelöscht werden soll und können in Ihrer Meldung Inhalte aus dieser Zeile anzeigen.

- RowDeleted: Dieses Ereignis wird nach der Löschung ausgeführt.

- BeforeNew: Dieses Ereignis wird dann ausgeführt, wenn der Benutzer auf die NEU-Schaltfläche geklickt hat, jedoch bevor irgend eine Aktion ausgeführt wird. Mit Cancel kann der Prozess abgebrochen werden.

- AfterNew: Dieses Ereignis wird nach dem Anlegen eines neuen Datensatzes ausgelöst, d.h. analog BeforeNew, jedoch dann, wenn alles bereits passiert ist. Dieses Ereignis ist insbesonders wertvoll, um den Index im Datalayer neu zu setzen, was nicht automatisch geschieht. Grundsätzlich geht das Control davon aus, dass der User mittels Klick den Index neu setzt, wenn jedoch weitere Informationen angezeigt werden sollen, dann muss der Index im RedMill DataLayer Objekt zuerst neu gesetzt werden.

5. Hinweise zum Zeilen und Spalten-Index

Das RedMill DataGrid Element beginnt den Index immer mit 1 (nicht mit 0!).
Beim Zeilenindex ist folgendes zu beachten, dass immer die Position in der gesamten Tabelle gemeint ist und nicht unbedingt die sichtbare Position.

Wenn Sie eine Tabelle mit 10 Datensätzen haben und sind mit dem Scrollbalken eine Zeile nach unten gefahren, sodass in der 9. sichtbaren Zeile der 10.Datensatz angezeigt wird, erhalten Sie als Index die 10 zurück und nicht die 9.

Der RedMill XML Handler (clsXML)

Der RedMill XML Handler ist eine zusätzliche Komponente (eigene DLL) vom RedMill DataLayer und arbeitet eng mit diesem zusammen, d.h. eine Arbeit ohne DataLayer ist nicht möglich.

Der RedMill XML Handler schreibt aus einem DataLayer-Objekt eine XML-Datei. Zudem ist der RedMill XML Handler in der Lage, eine XML-Datei zu lesen, sofern diese sich an bestimmte Regeln hält.

```
<?xml version="1.0" encoding="windows-1252" ?>
<Adressen>
<Recordset QUERY="Select * from adressen">
<record ID="1">
<id>1</id>
<Anrede>Herr</Anrede>
```

Am Beispiel erkennen Sie, dass das Root-Element (hier <Adressen>) gesetzt wurde. Das Root-Element kann einen beliebigen Inhalt haben. Beim Schreiben des Files müssen Sie das Element spezifizieren.

Als nächstes wird <Recordset QUERY=... angegeben. Diese Angabe wird automatisch geschrieben, wird beim Lesen jedoch dann benötigt, wenn Sie die zurückgegebene OfflineRecordsetKlasse in einem DataLayer-Objekt anhängen möchten.

Jeder einzelne Datensatz MUSS mit dem <record ... Tag gekennzeichnet werden. Ein anderer Tag wird nicht akzeptiert und ein Fehler über den ErrorStatus ausgegeben.

Natürlich müssen Sie sich auch an die XML-Regeln halten. Die HELP-Methode gibt Ihnen einen kurzen Hilfetext zurück.

1. Allgemein

   - XMLFilePath: Diese Write-Only Eigenschaft muss in jedem Fall gesetzt werden. Sie gibt an, wohin das XML-File geschrieben, resp. welches XML-File gelesen werden soll. Die Dateiendung XML ist nicht unbedingt nötig.
   - ErrorStatus: Diese Eigenschaft gibt allfällige Fehler zurück.

2. XML-Datei schreiben

   - OutputDataBinding: Beim Schreiben einer XML-Datei muss ein DataLayer-Objekt übergeben werden, welches die Daten beinhaltet. Es werden sämtliche Daten im DataLayer-Objekt in die XML-Datei geschrieben.
   - RootElement: Diese Eigenschaft bestimmt das Root-Element (im obigen Beispiel also <Adressen>. Dieses Element muss auf jeden Fall gesetzt werden.
   - WriteXML: Diese Methode schreibt das XML-File in dem Verzeichnis und Namen, welches mit der XMLFilePath Eigenschaft gesetzt wurde.

3. XML-Datei lesen

	ReadXML:	Diese Methode liest das XML-File im angegebenen Verzeichnis.
	RootElement:	Gibt das im XML-File vorkommende Root-Element wieder
	InputQuery:	Gibt das im XML-File vorkommende Query wieder (aus dem Tag <Recordset QUERY...
	InputData:	Mit dieser Eigenschaft können Sie das OfflineRecordset Objekt beziehen, welches die Daten aus dem XML-File beinhaltet. Das OfflineRecordset erhält die TAG-Eigenschaft automatisch korrekt gesetzt, damit dieses Objekt an einem DataLayer-Objekt angehängt werden kann.

Der RedMill Configuration Manager

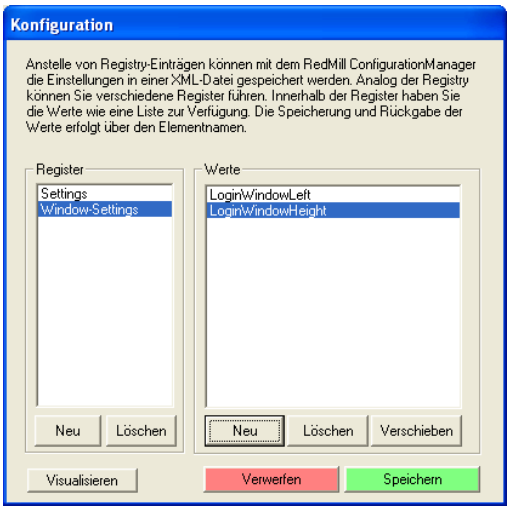
Der RedMill Configuration Manager (DLL) dient dazu, Konfigurations- und Einstellungs-Informationen, wie z.B. Benutzereinstellungen, analog der Registry zu speichern, jedoch in einer XML Datei. Der RedMill Configuration Manager hat ein visuelles Interface, wenn Sie Register und Werte manuell erstellen. So starten Sie das visuelle Interface:

```
Sub StartConfigurationManager()
Dim objConfiguration As New RedMillConfigManager.clsConfiguration
objConfiguration.DesignMode
End Sub
```

Sie können nun wählen, ob Sie eine neue Konfigurationsdatei erstellen oder eine bestehende bearbeiten möchten. Im Anschluss können Sie Register definieren sowie die einzelnen Konfigurationselemente.

Selbstverständlich kann auch alles ohne Design-Mode erstellt werden. Der Design-Mode greift auf die gleichen Eigenschaften und Methoden zu, welche Sie auch zur Verfügung haben.

Tipp: Es wird empfohlen, anstelle über die Methoden der nachfolgenden Rubriken 2 und 3, die Methoden GetConfigSetting und SetConfigSetting zu verwenden, da diese Mehtoden die anderen Methoden kapseln und sehr viel weniger Programmieraufwand nötig ist.



Das Register entspricht einem Registry-Schlüssel, unter diesem verschiedene Werte gespeichert werden. Hier können Sie die Register definieren, wie auch die Wertnamen. Die Werte selber speichern Sie über Methoden der Komponente.

1. **Arbeit mit Dateien**

   - CreateNewConfigFile:
     Geben Sie den vollen Pfad mit Dateinamen an. Dadurch wird eine neu Konfigurationsdatei geschrieben. Bitte beachten Sie, dass die Datei erst dann geschrieben wird, wenn Sie Register und Wertnamen erstellt haben.

   - OpenConfigFile: Öffnet eine XML-Konfigurationsdatei. Bitte beachten Sie, dass nur Dateien geöffnet werden können, welche mit dem RedMill ConfigurationManager erstellt wurden. Der Inhalt des Files wird ausgelesen und im Speicher als clsOfflineRecordset Objekt gehalten. Dieses Objekt kann für jedes Register (Record) verschiedene Anzahl von Feldern haben.

2. **Arbeit mit Registern**
   Register können beliebig verschiedene Wertnamen und zugeordnete Werte haben. Sie können auch beliebig verschiedene Register definieren. Allerdings bilden die Register eine einzige Hierarchie. Es bestehen keine Unterregister, wie dies in der Registry vorgesehen ist.

   - CreateNewRegister:
     Diese Methode Erstellt ein neues Register. Zudem müssen Sie mindestens einen WertNamen übergeben. Sie können verschiedene Wertnamen in einem String übergeben, wenn Sie die einzelnen Wertnamen mit einem Semikolon (;) abtrennen.

   - DeleteRegister: Löscht das Register und sämtliche untergeordnete Wertnamen und Werte, mittels Angabe des Registernamens.

- SetCurrentRegister:
  Setzt das Standardregister. Dieses wird bei der GetCurrentRegisterValue Methode benutzt, damit nicht jedes Mal das Register angegeben werden muss.

3. Arbeit mit Wertnamen

- CreateNewValueName:
  Unter Angabe von InRegister wird in dem angegebenen Register ein neuer Wertname angelegt, unter dem dann ein Wert gespeichert werden kann.

- DeleteConfigValueName:
  Löscht einen Wertnamen unter Angabe des Registers und des Wertnamens. Danach kann kein Wert mehr unter dem Wertnamen mehr gespeichert werden.

- SetConfigValue: Setzt einen Wert im Angegebenen Register und Wertnamen. Die Angabe von Register und Wertname muss mit einem Backslash (\) abgetrennt werden. Beispiel: `WindowsSettings\WindowLeft`

- ConfigValue: Gibt den Wert eines Elements zurück. Register\Wertname-Angabe analog SetConfigValue

- SaveCurrentRegisterValue:
  Diese Methode arbeitet analog der Methode "SetConfigValue", wobei das Register nicht angegeben werden muss, da dieses mit der Eigenschaft SetCurrentRegister gesetzt wird. Somit genügt die Angabe des Wertnamens und des zu speichernden Werts.

- GetCurrentRegisterValue:
  Diese Methode arbeitet analog der Eigenschaft "ConfigValue", wobei das Register nicht angegeben werden muss, da dieses mit der Eigenschaft SetCurrentRegister gesetzt wird. Somit genügt die Angabe des Wertnamens, um den darunter gespeicherten Wert zu ermitteln.

- HELP: Die HELP-Methode zeigt den Unterschied der verschiedenen hier beschriebenen Methoden und Eigenschaften.

Wie Sie den Ausführungen entnehmen, arbeitet die Komponente nicht analog der SaveSetting / GetSetting Methode von VB, welche die Werte in die Registry unter HKEY_CURRENT_USER\Software\VB and VBA Programs\.... speichert.

Das nachstehende Code-Beispiel zeigt jedoch zwei implementierte Methoden, welche genau das simulieren.

Der im Beispiel sichtbare Parameter ApplicationID dient lediglich dafür, dass Sie nicht überall im Code den Applikations-Schlüssel löschen müssen, wenn Sie von der VB-Funktion SaveSetting/GetSetting umstellen möchten. So können Sie über Suchen/Ersetzen einfach den Funktionsnamen von GetSetting in GetConfigSetting ändern und alles funktioniert weiter, d.h. Sie können den Schlüssel weiterhin übergeben, er wird einfach nicht verwendet, da der "Applikations-Schlüssel" ja der Pfad zur Konfigurationsdatei ist.

Wichtig ist einfach, dass beim Start der Applikation ein ConfigurationManager Objekt erzeugt wurde und dass das Konfigurationsfile geladen oder ein neues erzeugt wurde.

Beispielcode für die implementierten Methoden:

```
Function GetConfigSetting(ApplicationID As String, Register As String, Key
As String, Optional Default As String) As String

On Error GoTo fehler
Dim strResult As String
Dim lngErrorState As Long
strResult = objConfig.ConfigValue(Register & "\" & Key)
If strResult = "unbekannt" Or strResult = "" Then strResult = Default
GetConfigSetting = strResult
Exit Function

CreateRegister:
objConfig.CreateNewRegister Register, Key
GetConfigSetting = strResult
Exit Function

Register:
objConfig.SetCurrentRegister = Register
objConfig.CreateNewValueName Register, Key
GetConfigSetting = Default
Exit Function

fehler:
lngErrorState = lngErrorState + 1
If lngErrorState = 1 Then Resume Register
If lngErrorState = 2 Then Resume CreateRegister
End Function
```

Analog geht es auch mit der Function SaveConfigSetting:

```
Public Function SaveConfigSetting(ApplicationID As String, Register As
String, Key As String, ValueSetting As String)
On Error GoTo fehler
Dim strResult As String
Dim lngErrorState As Long
objConfig.SetConfigValue Register & "\" & Key, ValueSetting
Exit Function

CreateRegister:
objConfig.CreateNewRegister Register, Key
objConfig.SetConfigValue Register & "\" & Key, ValueSetting
Exit Function

Register:
objConfig.SetCurrentRegister = Register
objConfig.CreateNewValueName Register, Key
objConfig.SetConfigValue Register & "\" & Key, ValueSetting
Exit Function

fehler:
lngErrorState = lngErrorState + 1
If lngErrorState = 1 Then Resume Register
If lngErrorState = 2 Then Resume CreateRegister
End Function
```

Auf diese Art und Weise können Sie bequem in Ihrer Applikation Werte setzen und lesen, ohne dass Sie die Registry benötigen.
Wie gesagt, diese beiden Funktionen sind bereits implementiert.

Der RedMill File Manager

Die Komponente RedMillFileManager ist ein Dialogfeld, mittels welchem Sie eine Datei öffnen können. Natürlich wird nicht die Datei geöffnet sondern nur der Pfad zur gewählten Datei zurückgegeben. Der RedMill FileManager wird von verschiedenen Komponenten verwendet, daher wird diese Komponente ebenfalls mitgeliefert. Wenn Sie diese Komponente selber brauchen können, dann umso besser.

- DisplayFileManager:
  Die einzige Methode, welche die clsFileManager kennt, ist die "DisplayFileManager" Methode. Dadurch wird ein Formular angezeigt, mittels welchem der Benutzer eine Datei oder ein Ordner auswählen kann. Diese Methode wird immer als letztes aufgerufen, vorher müssen noch ein paar Eigenschaften gesetzt werden, danach können Eigenschaften abgerufen werden.

- strDefaultPath: Setzt den Vorgabepfad

- bolShowDirectoryOnly:
  Gibt an, ob ein Verzeichnis gewählt werden soll oder eine Datei.

- strFormCaption:
  Setzt den Titel des Formulars

- bolHideIcon: Versteckt das RedMill-Icon

- strAlternativeIcon:
  Mittels Angabe des Pfads und Iconname in einem String, kann ein alternatives Icon gesetzt werden.

Nach dem Schliessen des Formulars werden die folgenden Eigenschaften gesetzt, welche danach abgefragt werden können:

- bolCancelled: True, wenn der Benutzer die Aktion abgebrochen hat

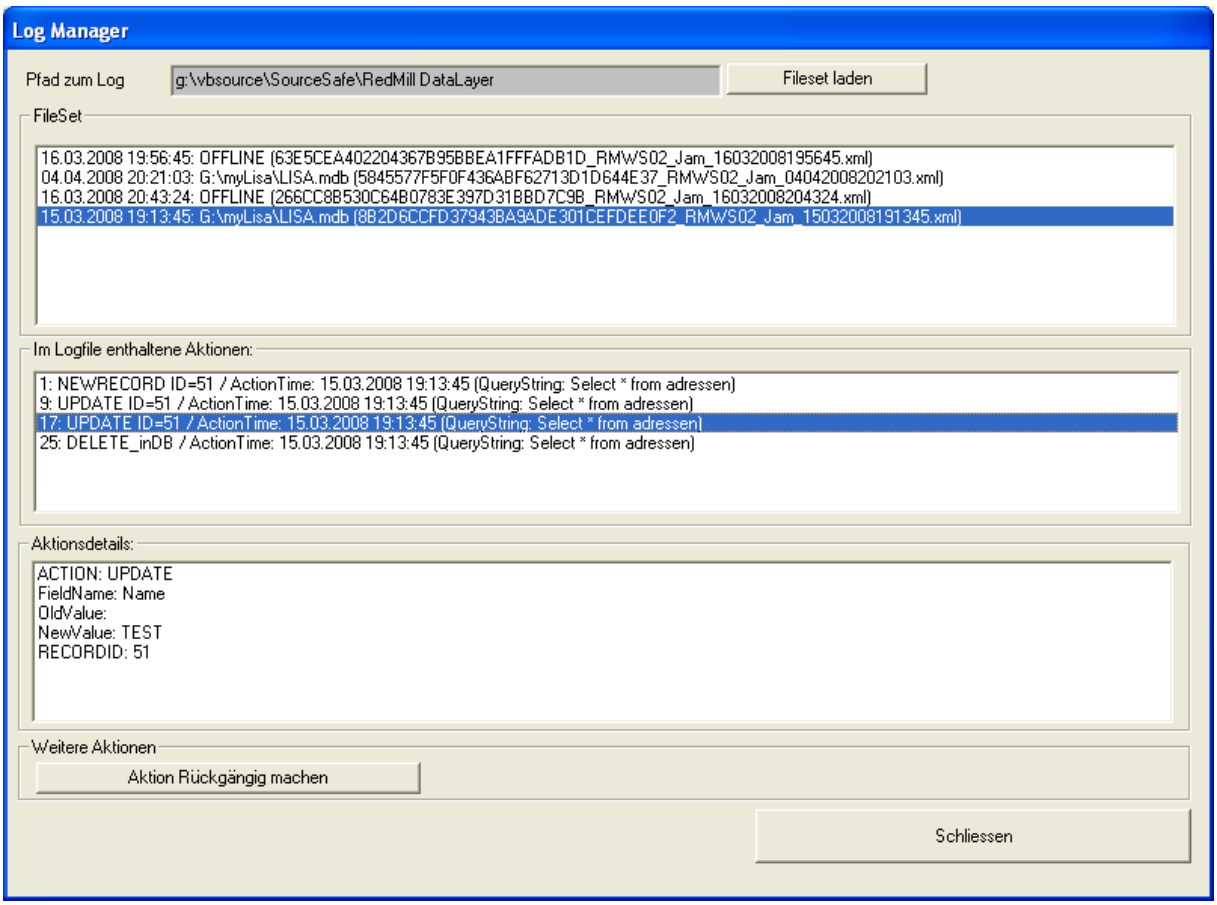
- strResultFullPath:
  Gibt den genauen Pfad inkl. Dateiname zurück, d.h. von der Datei, welche der Benutzer ausgewählt hat.

- strResultFileName:
  Gibt nur den ausgewählten Dateinamen zurück

Der RedMill Log Manager

Die DataLayer Komponente schreibt für jede Instanz ein eigenes Logfile, wenn Daten verändert wurden. Dieses Logfile kann mit dem RedMill LogManager bearbeitet werden. Am einfachsten ist der Aufruf des visuellen Interfaces:

```
Sub DisplayLogManager()
    Dim objLog As New RedMillLogManager.LogManager
    objLog.Display
End Sub
```



Im Wesentlichen greift dieses Interface auf die Logik der einzelnen Klassen zurück, welche nachfolgend beschrieben werden:

1. **LogManager Klasse**

   - Display: Zeigt das obige Interface an
   - EjectFileSet: Setzt das bestehende FileSetObjekt = nothing
   - FileSetManager
     Gibt das FileSetManager Objekt zurück. Wenn dieses noch nicht vorhanden ist, wird ein neues Objekt erzeugt.

2. FileSetManager Klasse

- EjectFileSet: Setzt das bestehende FileSetObjekt = nothing
- LoadFileSet: Lädt ein Fileset (alle Logdateien in einem anzugebenden Verzeichnis [Path])
- ReadFileSetProperties: Lädt alle Eigenschaften aus allen XML-Dateien.
- File Gibt den Dateinamen des Files gemäss [Index] zurück.
- FileCount Gibt die Anzahl der Files im FileSetObjekt zurück
- FileProperties: Gibt die Eigenschaft zurück aus dem File [Index] und angegebenen [PropertyType]. Es werden die folgenden Property Types unterstützt:
  SESSIONID
  SESSIONTIME
  DATABASE
  COMPUTERNAME
  SYSTEMUSERNAME
  LICENCEUSERNAME
  DECLAREDUSERNAME
- LogContent Gibt ein LogContent Objekt zurück, welches die Logdaten enthält. Angabe des [FileIndex]
- DeleteOldFiles: Löscht alle Logfiles, welche älter sind als das angegebene Datum.

3. LogContent Klasse

- LoadLogData: Lädt die Aktionen aus dem Logfile
- GetActionList: Gibt ein CollectionObjekt zurück mit den Aktionen. In dieser Liste ist der Handle mit einem Doppelpunkt abgetrennt zur Aktion im File. Der Handle entspricht der Position der Aktion im XML-File (Zeile).
- GetActionData: Mit Übergabe des Aktions-Handles erhält man ein PropertyCollection Objekt, in welchem die Aktionen enthalten sind.
- ErrorInfo Sind im DataLayer Fehler entstanden, werden diese auch im Logfile vermerkt und können mit dieser Eigenschaft abgerufen werden.
- Parent Zugriff auf das Parent-Objekt
- FileIndex FileHandle für interne Zwecke
- RetrieveActionQueryString: Der SQL-Query, welche dem DataLayer mit GetRecordset übergeben wurde, wird ebenfalls geloggt.
- ResetErrors Löscht die Fehlerinformation

4. Property Collection Klasse

Die PropertyCollection Klasse dient zum Sammeln und Aufbewahren von Informationen. Es werden intern 2 Collection-Objekte verwendet, welche kongruent gepflegt werden.

- AddProperty — Mit der Übergabe des Property-Namens und des Werts (String) wird ein neuer Eintrag angelegt.
- PropertyCount — Gibt die Anzahl der Einträge zurück
- PropertyContent — Mit Übergabe des PropertyName wird der Inhalt des unter dem Namen gespeicherten Eintrags zurückgegeben.
- PropertyNameByIndex — Gibt den PropertyName gemäss dem Eintrag-Index zurück.
- PropertyValueByIndex — Gibt den Inhalt des Speichereintrags gemäss Index zurück.

5. ActionHandler Klasse

Die ActionHandler Klasse hat zur Aufgabe, die im Logfile enthaltenen Aktionen rückgängig zu machen oder diese nochmals durchzuführen. Letzteres wird in einer späteren Version nach Bedarf implementiert, d.h. es ist momentan nur die Rückgängig-Methode implementiert.

Natürlich ist es nur möglich, Aktionen rückgängig zu machen, welche im DataLayer erfolgt sind. Datensätze, welche innerhalb der Datenbank aufgrund einer Löschweitergabe gelöscht wurden, werden nicht geloggt und können daher auch nicht rückgängig gemacht werden.

Es ist nur möglich, Aktionen rückgängig zu machen, wenn mit einer Datenbank gearbeitet wurde. Nur dann werden auch alle Aktionen geloggt, welche datenbankrelevant sind. Ohne Datenbankanbindung wird im DataLayer ja auch die Save-Methode nicht verwendet und daher werden Änderungen auch nicht geloggt. Es werden nur Änderungen geloggt, wenn diese datenbankrelevant sind. Eine Ausnahme bildet die Anweisung RemoveRecord, welche vom DataLayer auch dann geloggt wird, wenn in einer Datenbank nichts verändert werden soll oder wenn nicht mit einer Datenbank gearbeitet wird.

- FileSetObjekt — Übergabe des Fileset Objekts
- LogFileHandle — Index des zu verwendenden Logfiles
- ActionHandle — Übergabe des Actionhandle
- UndoAction — Macht die Aktion im FileSetObjekt gemäss LogfileIndex und ActionIndex rückgängig. Sprich: Es wird eine DataLayer-Instanz erzeugt und die GetRecordset Methode mit dem original SQL-Query aufgerufen. Dann wird der Datensatz mit der ID aufgerufen und der geloggte alte Wert wieder gesetzt. Dies schreibt natürlich wieder ein neues Logfile.

Deployment

Neben der RedMill DataLayer Komponente müssen Sie (sofern verwendet) die folgenden Komponenten mitliefern:

Wenn Sie Dateien in eine XML-Datei schreiben und diese wieder lesen wollen
Zwingend: RedMill DataLayer, RedMill XML Handler

Wenn Sie den Configuration Manager verwenden möchten:
Zwingend: RedMill Configuration Manager, RedMill File Manager, RedMill XMLHandler, RedMill DataLayer

Wenn Sie mit einem Steuerelement arbeiten:
RedMillDataGridControl und RedMill DataLayer und/oder
RedMillFormDataControl und RedMill DataLayerl

Wenn Sie die Logs visualisieren möchten:
RedMill LogManager und RedMill DataLayer

Kontakt

RedMill
Alexander Mühle
Kehrsitenstr. 7
6362 Stansstad
Schweiz
http://software.redmill.ch
squirrel@redmill.ch