

OFFICE TALK

Kommunikationsbibliothek

*Active Data Objects(ADO)
for OfficeTalk*

Version 2.00

Inhaltsverzeichnis

| | |
|---|----|
| Einleitung | 1 |
| Voraussetzungen | 1 |
| Unterstützte Datenbanksysteme | 1 |
| Kommunikationsbibliothek | 2 |
| Allgemeine Syntax | 2 |
| Dienst Syntax | 2 |
| Fehler bei der Dienstausführung | 2 |
| Datenbankspezifische Verbindung | 3 |
| Dienste | 4 |
| beginTrans | 5 |
| Syntax | 5 |
| Bemerkung | 5 |
| Beispiel | 5 |
| cancelBatch | 5 |
| Syntax | 5 |
| Beispiel | 5 |
| close | 6 |
| Syntax | 6 |
| Beispiel | 6 |
| commitTrans | 7 |
| Syntax | 7 |
| Bemerkung | 7 |
| Beispiel | 7 |
| eof | 7 |
| Syntax | 7 |
| Beispiel | 7 |
| execute | 8 |
| Syntax | 8 |
| Bemerkung | 8 |
| Beispiel | 8 |
| fields | 9 |
| Syntax | 9 |
| Beispiel | 9 |
| item | 9 |
| Syntax | 9 |
| Beispiel | 10 |
| moveFirst | 10 |
| Syntax | 10 |
| Beispiel | 10 |
| moveNext | 11 |
| Syntax | 11 |
| Beispiel | 11 |
| open | 11 |
| Syntax | 11 |
| Bemerkung | 12 |
| Beispiel-1 | 12 |
| Beispiel-2 | 12 |
| rollbackTrans | 12 |
| Syntax | 12 |
| Bemerkung | 13 |
| Beispiel | 13 |
| updateBatch | 13 |
| Syntax | 13 |
| Beispiel | 13 |
| Beispiele | 15 |
| Datensätze suchen, lesen und weiter verwenden | 15 |

| | |
|---------------------------|----|
| Beispiel..... | 15 |
| Datensatz schreiben | 16 |
| Beispiel..... | 16 |
| Datensatz ändern | 17 |
| Beispiel..... | 17 |
| Datensatz löschen | 18 |
| Beispiel-1..... | 18 |
| Beispiel-2..... | 19 |

Einleitung

Diese Dokumentation beschreibt die Anwendung der Dienste zur Kommunikation mit beliebigen Datenbankservern im Rahmen von Skripts und Skriptmakros. Damit werden Sie in die Lage versetzt, mit OfficeTalk Tabellen in einer Datenbank zu bearbeiten. Wie Sie Skripts und Skriptmakros erstellen, ändern und löschen, lesen Sie in der Dokumentation [OfficeTalk Business-Process-Management](#) und [OfficeTalk Skript](#).

Voraussetzungen

Um die Kommunikationsbibliothek für SQL-Server Datenbanken zu benutzen, müssen folgende Voraussetzungen gegeben sein:

- OfficeTalk wird auf einer Windows Plattform betrieben.
- Das Datenbanksystem unterstützt die ADO-Funktionalität.
- Die ActiveX Komponente **Microsoft ActiveX DataObjects 2.7 Library** ist installiert. Die Komponente wird durch die Systemdatei **D:\Programme\Gemeinsame Dateien\System\msado15.dll** repräsentiert. I.d.R. wird diese Komponente bei der Installation von Windows automatisch mit installiert. Falls die Datei auf Ihrem Rechner trotzdem fehlt, können Sie die Komponente durch die Installation des **MDAC (Microsoft Data Access Components)** installieren. **MDAC** erhalten Sie von <http://msdn.microsoft.com/data/ref/mdac/downloads> als Download.

Hinweis: Daß die Komponente auf Ihrem Rechner fehlt, erkennen Sie an der Fehlermeldung **Klasse <Name der Klasse> ist nicht registriert** bei der Erzeugung eines neuen Datenbankobjektes (z.B. `SmallCOM.ADODB.Connection New`) in einem Makro.

Unterstützte Datenbanksysteme

Folgende Datenbanksysteme werden mindestens unterstützt:

- Microsoft-SQL-Server
- IBM-DB2-Server
- Oracle-SQL-Server
- PostgreSQL-Server
- Microsoft-Access
- Microsoft Foxpro
- Borland Delphi

Hinweis: Ob ein Datenbanksystem mit dieser Kommunikationsbibliothek unterstützt wird, hängt von der Verfügbarkeit eines ADO-Treibers für das Datenbanksystem ab. Lesen Sie dazu die Dokumentation des Datenbanksystems.

Kommunikationsbibliothek

Um die Dienste im Skriptmakro anzuwenden, müssen Sie die zugehörige Bibliothek als erstes mit der Anweisung `Library ..\Library\Microsoft ADO DB.pcl` laden. Wenn sich die Bibliothek in einem anderen Verzeichnis befindet, ändern Sie den Pfad entsprechend ab.

Allgemeine Syntax

Die generelle Syntax ist in der Dokumentation [OfficeTalk Skript](#) nachzulesen. Hier wird nur die Syntax, bezogen auf die Dienste der Kommunikationsbibliothek, beschrieben.

Dienst Syntax

In diesem Kapitel wird die allgemeine Syntax der Bibliotheksdienste beschrieben. Die Syntax entspricht weitgehend der Syntax von Microsoft® VisualBasic. Nachfolgend sind Anweisungen oder Anweisungsteile im Gegensatz zu den übrigen Texten in der Schriftart `Courier` geschrieben. Die Schreibweise der folgenden Syntax:

- `<` und `>` sind zu ersetzen durch den aktuellen Namen, Bezeichner oder ähnlichem.
- `[` und `]` umschließen optionale Angaben und werden nicht geschrieben.
- `...` kennzeichnen weitere gleichartige Angaben und werden nicht geschrieben.
- Mehrere Angaben durch `|` getrennt sind alternativ und mit `{ }` umschlossen.
- Wörter ohne obige Kennzeichnung sind unverändert zu übernehmen.

Fehler bei der Dienstausführung

Wenn Sie nicht sicher sind, ob der Dienst vom Datenbankserver akzeptiert wird, sollten Sie ihn in einen Try-Catch Block kapseln, da das Skriptmakro sonst bei einem Ausführungsfehler abbricht. In dem Catch Teil können Sie dann pragmatisch auf den Fehler reagieren. Ein Try-Catch Block entspricht der VisualBasic Anweisung `On Error GoTo <Label>`, ist jedoch übersichtlicher. Weiteres dazu lesen Sie in der Dokumentation [OfficeTalk Skript](#). Ein kleines Beispiel dazu:

```
Library "..\Library\Microsoft ADO DB.pcl"
```

```
Dim rs As SmallCOM.ADO DB.Recordset
```

```
rs = New SmallCOM.ADO DB.Recordset
```

```
Try
```

```
    rs.open( "select * from Adresse",
```

```

activeConnection: "Driver={SQL Server};

Server=DBServer;Uid=sa;Pwd=; Database=pubs" )

End Try

Catch

MsgBox("Datenbank-Open ist fehlgeschlagen.<n>" & "Fehler: " & Er-
ror.Description & "<n> Bitte wiederholen")

'open mit korrigierten Argumenten wiederholen

End Catch

```

Datenbankspezifische Verbindung

Entsprechend des zu verwendenden Datenbanksystems müssen Sie im Argument `activeConnection`: der ersten *open*-Anweisungsvariante oder im ersten Argument der zweiten *open*-Anweisungsvariante die datenbankspezifischen Verbindungsanweisungen schreiben. Grundsätzlich stehen drei Arten der Verbindungsanweisungen zur Verfügung:

1. **Driver**: Die Verbindungsanweisungen werden in der **Driver**-Syntax angegeben. In der nachfolgenden Tabelle finden Sie einige Beispiele dazu. Einzelheiten zu weiteren Verbindungsanweisungen im Driverformat finden Sie in der Dokumentation zu *Microsoft-ADO* und zur Datenbank.
2. **Provider**: Die Verbindungsanweisungen werden in der **Provider**-Syntax angegeben. In der nachfolgenden Tabelle finden Sie einige Beispiele dazu. Einzelheiten zu weiteren Verbindungsanweisungen im Providerformat finden Sie in der Dokumentation zu *Microsoft-ADO* und zur Datenbank.
3. **DSN**: Die Verbindungsanweisungen werden in der **DSN**-Syntax angegeben. Mit der Startleiste **Einstellungen - Systemsteuerung - Verwaltung - ODBC-Datenquelle** können Sie DSN-Einträge erstellen. In der nachfolgenden Tabelle finden Sie einige Beispiele dazu. Einzelheiten zu weiteren Verbindungsanweisungen im DSN-Format finden Sie in der Dokumentation zu *Microsoft-DSN* und zur Datenbank.

| Datenbank | Verbindungsanweisung |
|---|--|
| Driver | Driver="<Verbindungsanweisung>" |
| Microsoft-SQL-Server oder PostgreSQL | <Verbindungsanweisung>: <System>;Server=<Server>;Uid=<Benutzer>;Pwd=<Passwort>; Database=<Datenbank>; <System>: Das Datenbanksystem { SQL server }: Microsoft-SQL-Server { PostgreSQL Unicode }: PostgreSQL-Server <Server>: Der Netzwerknamen des Datenbankservers <Benutzer>: Der Loginname für die Datenbank <Passwort>: Das Passwort für die Datenbank <Datenbank>: Der Datenbankname |
| Foxpro | <Verbindungsanweisung> |

| Datenbank | Verbindungsanweisung |
|------------------|---|
| | <pre>Driver={Microsoft Visual FoxPro Driver}; SourceType= <Dateiformat>; SourceDb=<Datenbank>;</pre> <p><Dateiformat>: Das Dateiformat der Datenbank</p> <p>DBF-Datenbank ist in DBF-Dateien enthalten.</p> <p>DBC-Datenbank ist in DBC-Dateien enthalten.</p> <p><Datenbank>: Das Verzeichnis mit den Datenbankdateien</p> |
| Provider | Provider="<Verbindungsanweisung>" |
| Microsoft-Access | <pre><Verbindungsanweisung> Microsoft.Jet.OLEDB.4.0;Data Source= <Datenbank>;</pre> <p><Datenbank> Der Dateiname der mdb-Datenbank</p> |
| IBM-DB2 | <pre><Verbindungsanweisung> IBMDADB2;HOSTNAME=<Server>;Database=<Datenbank>;PROTOCOL= TCPIP;PORT=50000;uid=<Benutzer>;pwd=<Passwort>;</pre> <p><Server>: Der Netzwerknamen des Datenbankservers</p> <p><Datenbank>: Der Name der Datenbank</p> <p><Benutzer>: Der Loginname für die Datenbank</p> <p><Passwort>: Das Passwort für die Datenbank</p> |
| Komponente | <p>Der Name der Komponente für den Datenbankzugriff. Das ist i.d.R. der Name einer registrierten DLL-Datei. Zur Verwendung kommt dabei das Provider-Format.</p> <p>"Provider=<Verbindungsanweisung>"</p> |
| Foxpro-Beispiel | <pre><Verbindungsanweisung> vfpoledb; Data Source= <Datenbank></pre> <p><Datenbank>: Das Verzeichnis mit den Datenbankdateien</p> |
| DSN | <pre><Verbindungsanweisung></pre> <p>Der Name des System- oder User DSN-Eintrages aus den ODBC-Datenquellen. Der Eintrag enthält die weiteren Informationen zur Verwendung der Datenbank.</p> |

Hinweis: Für die Bearbeitung vorgangsspezifischer Schemata muss ein separater Datenbankbenutzer mit den entsprechenden Rechten für das Schema angelegt werden. Ein Bearbeiters aus OfficeTalk kann dafür nicht verwendet werden, weil das Passwort eines Bearbeiters nicht das eingetippte, sondern ein daraus abgeleiteter Fingerprint ist. Im einfachsten Fall werden der Datenbankgruppe OFFICETALKUSER die Rechte für das fachspezifische Schema gegeben und der Datenbankbenutzer als Mitglied der Datenbankgruppe OFFICETALKUSER eingetragen.

Dienste

In den nachfolgenden Kapiteln finden Sie nur die wichtigsten Dienste und einige Beispiele dazu. Die Beschreibung der weiteren Datentypen/Klassen und Dienste entnehmen Sie bitte der Doku-

mentation *Microsoft ActiveX DataObjects*. Die Namen der Datentypen im Skriptmakro beginnen mit dem Präfix SmallCOM.ADODB.

beginTrans

Der Dienst startet eine neue Transaktion.

Syntax

```
beginTrans
```

Bemerkung

Der Dienst ist nur bei Verwendung einer Connection möglich. Bei Verwendung eines Recordset siehe Dienst `updateBatch` und `cancelBatch`.

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
con = New SmallCOM.ADODB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs",
```

```
userID: "sa")
```

```
con.beginTrans
```

cancelBatch

Der Dienst verwirft die Änderungen in einen Recordset. Der Dienst kann nur in Verbindung mit einem Recordset angewandt werden, und verhält sich ähnlich wie der Dienst `rollbackTrans`.

Syntax

```
cancelBatch
```

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
Dim rs As SmallCOM.ADODB.Recordset
```

```
con = New SmallCOM.ADODB.Connection  
con.open("Driver={SQL Server};Server=DBServer;Database=pubs;Uid=sa")  
rs = con.execute( "SELECT * FROM titles" )  
While rs.eof = FALSE  
  ' Tu etwas  
rs.MoveNext  
Wend  
rs.cancelBatch  
rs.close
```

close

Der Dienst schließt die geöffnete Verbindung oder das geöffnete Recordset.

Syntax

```
close
```

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
Dim rs As SmallCOM.ADODB.Recordset
```

```
con = New SmallCOM.ADODB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs",
```

```
userID: "sa")
```

```
rs = con.execute( "SELECT * FROM titles" )
```

```
rs.close
```

```
con.close
```

commitTrans

Der Dienst schließt die laufende Transaktion. Alle bisherigen Änderungen werden in der Datenbank dauerhaft gemacht.

Syntax

```
commitTrans
```

Bemerkung

Der Dienst ist nur bei Verwendung einer Connection möglich. Bei Verwendung eines Recordset siehe Dienst updateBatch.

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
Dim rs As SmallCOM.ADODB.Recordset
```

```
con = New SmallCOM.ADODB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs", userID: "sa")
```

```
con.beginTrans
```

```
rs = con.execute( "DELETE FROM titles" )
```

```
con.commitTrans
```

```
rs.close
```

eof

Der Dienst meldet, ob das Recordset am letzten Datensatz angekommen ist. TRUE bedeutet Ende erreicht. FALSE bedeutet Ende nicht erreicht.

Syntax

```
eof
```

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim rs As SmallCOM.ADODB.Recordset
```

```
rs = New SmallCOM.ADO.DB.Recordset

rs.open( "SELECT * FROM titles", activeConnection:
        "Driver={SQL Server}; Server=DBServer;Uid=sa;Pwd=;Database=pubs"

While rs.eof = FALSE

    ' Tu etwas

    rs.MoveNext

Wend

rs.close.
```

execute

Mit dem Dienst wird eine SQL-Anweisung ausgeführt. Bei SELECT Anweisungen kann das Ergebnis als Recordset weiterverarbeitet werden.

Syntax

```
execute( <SQL-Anweisung> )
```

SQL-Anweisung benennt die SQL Anweisung oder SQL-Anweisungen.

<Ausdruck | Funktion | Literal | Variable>

Bemerkung

Der Dienst liefert i.d.R. ein Recordset zurück. Änderungen können zur Sicherheit in eine Transaktion gekapselt werden.

Beispiel

```
Library "..\Library\Microsoft ADO.DB.pcl"
```

```
Dim con As SmallCOM.ADO.DB.Connection
```

```
Dim rs As SmallCOM.ADO.DB.Recordset
```

```
con = New SmallCOM.ADO.DB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs", userID: "sa")
```

```
con.beginTrans
```

```
con.execute( "SELECT * FROM titles" )
```

```
con.execute( "DELETE FROM Adresse WHERE name = 'Müller'" )
```

```
If MsgBox("Änderungen speichern", , vbYesNo) = 1 Then  
    con.commitTrans  
Else  
    con.rollbackTrans  
End `If
```

fields

Der Dienst liefert alle Spalten des aktuellen Records aus einem Recordset.

Syntax

fields

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"  
  
Dim con As SmallCOM.ADODB.Connection  
Dim rs As SmallCOM.ADODB.Recordset  
Dim fields As SmallCOM.ADODB.Fields  
  
con = New SmallCOM.ADODB.Connection  
con.open("Driver={SQL Server};Server=DBServer;Database=pubs", userID: "sa")  
rs = con.execute( "SELECT * FROM titles" )  
fields = rs.fields
```

item

Der Dienst liefert eine benannten Spalte aus einer Fields-Sammlung, den Spalten des Recordsets. Der Wert der Spalte wird mit dem Dienst value geliefert.

Syntax

```
item( <Spaltenname> )
```

Spaltenname benennt die Spalte des Records.
<Ausdruck | Funktion | Literal | Variable>

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
Dim rs As SmallCOM.ADODB.Recordset
```

```
Dim fields As SmallCOM.ADODB.Fields
```

```
Dim field As SmallCOM.ADODB.Field
```

```
con = New SmallCOM.ADODB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs", userID: "sa")
```

```
rs = con.execute( "SELECT * FROM titles" )
```

```
fields = rs.fields
```

```
field = fields.item( "notes" )
```

```
MsgBox(field.value)
```

```
rs.close
```

moveFirst

Der Dienst positioniert das Recordset auf den ersten Datensatz.

Syntax

```
moveFirst
```

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
Dim rs As SmallCOM.ADODB.Recordset
```

```
con = New SmallCOM.ADODB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs;Uid=sa")
```

```
rs = con.execute( "SELECT * FROM titles" )
```

```
rs.moveFirst
```

moveNext

Der Dienst positioniert im Recordset auf den nächsten Datensatz.

Syntax

```
moveNext
```

Beispiel

```
Library "..\Library\Microsoft ADO DB.pcl"
```

```
Dim con As SmallCOM.ADO DB.Connection
```

```
Dim rs As SmallCOM.ADO DB.Recordset
```

```
con = New SmallCOM.ADO DB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs;Uid=sa")
```

```
rs = con.execute("SELECT * FROM titles")
```

```
While rs.eof = FALSE
```

```
    ' Tu etwas
```

```
    rs.moveNext
```

```
Wend
```

```
rs.close.
```

open

Der Dienst öffnet eine Datenbankverbindung oder ein Recordset mit dem Ergebnis einer SQL-Anweisung.

Syntax

1. `open(<SQL Anweisung>, activeConnection: <SQL-Verbindung>)`
SQL-Anweisung benennt die SQL Anweisung als <Ausdruck | Funktion | Literal | Variable>
SQL-Verbindung benennt die SQL Verbindungsoptionen als <Ausdruck | Funktion | Literal | Variable>
2. `open(<SQL-Verbindung>)`
SQL-Verbindung benennt die SQL Verbindungsoptionen als <Ausdruck | Funktion | Literal | Variable>

Für die Beschreibung der Verbindungsoptionen siehe Kapitel *Datenbankspezifische Verbindung*.

Bemerkung

Das erste Anweisungsformat verwenden Sie, um ein Recordset mit dem Ergebnis der SQL-Anweisung zu öffnen (Beispiel-1). Das zweite Anweisungsformat verwenden Sie, um eine Connection zu öffnen (Beispiel-2).

Die Zeichenkette mit den datenbankspezifischen Verbindungsinformationen kann auch von einem separaten Makro stammen. Schreiben Sie als Argument `activeConnection:` die Call-Anweisung mit dem Makro. z.B.: `open(<SQL Anweisung>, activeConnection: Call DB.DB2Connection)`. Dieses Makro muss mit einer Return-Anweisung enden. Das Argument der Return-Anweisung sind die Verbindungsinformationen als String.

z.B.: `Return "Driver={SQL Server};Server=DBServer;Database=pubs;Uid=sa"`

Beispiel-1

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim rs As SmallCOM.ADODB.Recordset
```

```
rs = New SmallCOM.ADODB.Recordset
```

```
rs.open( "SELECT * FROM Adresse",activeConnection:  
        "Driver={SQL Server}; Server=DBServer;Uid=sa;Pwd=; Database=pubs" )
```

Beispiel-2

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
con = New SmallCOM.ADODB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs", userID: "sa")
```

rollbackTrans

Der Dienst beendet die laufende Transaktion und verwirft alle seit `beginTransaction` erfolgten Datenbankänderungen.

Syntax

```
rollbackTrans
```


Bemerkung

Der Dienst ist nur bei Verwendung einer Connection möglich. Bei Verwendung eines Recordset siehe Dienst cancelBatch.

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
con = New SmallCOM.ADODB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs", userID: "sa")
```

```
con.beginTrans
```

```
con.execute( "DELETE FROM titles" )
```

```
con.rollbackTrans
```

updateBatch

Der Dienst macht die Änderungen in einen Recordset dauerhaft. Der Dienst kann nur in Verbindung mit einem Recordset angewandt werden, und verhält sich ähnlich wie der Dienst commitTrans.

Syntax

```
updateBatch
```

Beispiel

```
Library "..\Library\Microsoft ADODB.pcl"
```

```
Dim con As SmallCOM.ADODB.Connection
```

```
Dim rs As SmallCOM.ADODB.Recordset
```

```
con = New SmallCOM.ADODB.Connection
```

```
con.open("Driver={SQL Server};Server=DBServer;Database=pubs;Uid=sa")
```

```
rs = con.execute( "SELECT * FROM titles" )
```

```
While rs.eof = FALSE
```

```
    ' Tu etwas
```

```
rs.moveNext
```

```
Wend
```

```
rs.updateBatch
```

```
rs.close
```

Beispiele

Dieses Kapitel zeigt die wichtigsten Datenbankaktivitäten anhand von Beispielen. Da Datenbankaufgaben sehr weitgefächert sind, können diese Beispiele nur Ausschnitte aufzeigen.

Datensätze suchen, lesen und weiter verwenden

Das Beispiel zeigt, wie ausgewählte Datensätze aus einer Tabelle gelesen und der Inhalt weiter verwendet wird. Das Beispiel geht von folgenden Annahmen aus:

- Der Rechnername des Datenbankservers lautet DBServer.
- Das Login ist sa, kein Passwort.
- Die Datenbank heisst pubs.
- Die Tabelle heisst Adresse und hat die Spalten Name, Plz und Ort.

Die Adressen deren Spalte Name mit dem Buchstaben M beginnt, sollen in einer MessageBox ausgegeben werden.

Beispiel

AdresseSuchen

```
Library "..\Library\Microsoft ADO DB.pcl"

Dim recordset As SmallCOM.ADO DB.Recordset

Dim fields As SmallCOM.ADO DB.Fields

Dim field As SmallCOM.ADO DB.Field

Dim name As String

Dim plz As String

Dim ort As String

Dim text As Array

recordset = New SmallCOM.ADO DB.Recordset

recordset.cursorLocation( Enum( SmallCOM.ADO DB. CursorLocationEnum,
                                adUseClient ) )

recordset.lockType( Enum( SmallCOM.ADO DB.LockTypeEnum,
                          adLockBatchOptimistic ) )
```

```
recordset.open( "select * from Adresse where Name LIKE 'M%'",  
    activeConnection:  
        "Driver={SQL Server};Server=<Server>;Uid=sa;Pwd=;Database=pubs" )  
  
While recordset.eof = FALSE  
  
    fields = recordset.fields  
  
    field = fields.item( "Name" )  
  
    name = field.value  
  
    field = fields.item( "Plz" )  
  
    plz = field.value  
  
    field = fields.item( "Ort" )  
  
    ort = field.value  
  
    text = Array("Die Firma ",name," ist in ",plz," ",ort," beheimatet")  
  
    MsgBox( text )  
  
    recordset.moveNext  
  
Wend  
  
recordset.close
```

Datensatz schreiben

Das Beispiel zeigt, wie ein neuer Datensatz in eine vorhandene Tabelle geschrieben wird. Das Beispiel geht von folgenden Annahmen aus:

- Der Rechnername des Datenbankservers lautet DBServer.
- Das Login ist sa, kein Passwort.
- Die Datenbank heisst pubs.
- Die Tabelle heisst Adresse und hat die Spalten Name, Plz und Ort.

Die Adresse Müller, D-81739, München soll neu eingetragen werden.

Beispiel

AdresseSchreiben

```
Library "..\Library\Microsoft ADO DB.pcl"  
  
Dim recordset As SmallCOM.ADO DB.Recordset  
  
Dim fields As SmallCOM.ADO DB.Fields  
  
Dim field As SmallCOM.ADO DB.Field
```

```
recordset = New SmallCOM.ADODB.Recordset

recordset.cursorLocation( Enum( SmallCOM.ADODB. CursorLocationEnum,
                                adUseClient ) )

recordset.lockType( Enum( SmallCOM.ADODB.LockTypeEnum,
                          adLockBatchOptimistic ) )

recordset.open( "select * from Adresse", activeConnection:
               "Driver={SQL Server};Server=DBServer;Uid=sa; Pwd=;Database=pubs" )

recordset.addNew

fields = recordset.fields

field = fields.item( "Name" )

field.value( "Müller" )

field = fields.item( "Plz" )

field.value( "D-81739" )

field = fields.item( "Ort" )

field.value( "München" )

recordset.updateBatch

recordset.close
```

Datensatz ändern

Das Beispiel zeigt, wie ein bestehender Datensatz in einer Tabelle geändert wird. Das Beispiel geht von folgenden Annahmen aus:

- Der Rechnername des Datenbankservers lautet DBServer.
- Das Login ist sa, kein Passwort.
- Die Datenbank heisst pubs.
- Die Tabelle heisst Adresse und hat die Spalten Name, Plz und Ort.

Die Spalte Plz des Datensatzes Müller soll nach D-81700 geändert werden.

Beispiel

AdresseÄndern

```
Library "..\Library\Microsoft ADODB.pcl"

Dim recordset As SmallCOM.ADODB.Recordset

Dim fields As SmallCOM.ADODB.Fields
```

```
Dim field As SmallCOM.ADODB.Field

recordset = New SmallCOM.ADODB.Recordset

recordset.cursorLocation( Enum( SmallCOM.ADODB. CursorLocationEnum,
                                adUseClient ) )

recordset.lockType( Enum( SmallCOM.ADODB.LockTypeEnum,
                          adLockBatchOptimistic ) )

recordset.open( "select * from Adresse where Name = 'Müller'",
               activeConnection:
               "Driver={SQL Server};Server=DBServer;Uid=sa;Pwd=; Database=pubs" )

If recordset.recordCount >= 1

Then

    fields = recordset.fields

    field = fields.item( "Plz" )

    field.value( "D-81700" )

    recordset.updateBatch

End If

recordset.close
```

Datensatz löschen

Das Beispiel zeigt, wie ein bestehender Datensatz in einer Tabelle gelöscht wird. Im ersten Beispiel wird der Datensatz in ein Recordset gelesen und gelöscht. Im zweiten Beispiel wird der Datensatz durch die SQL-Anweisung `DELETE` gelöscht. Das Beispiel geht von folgenden Annahmen aus:

- Der Rechnername des Datenbankservers lautet `DBServer`.
- Das Login ist `sa`, kein Passwort.
- Die Datenbank heisst `pubs`.
- Die Tabelle heisst `Adresse` und hat die Spalten `Name`, `Plz` und `Ort`.

Der Datensatz, dessen Spalte `Name` den Wert `Müller` hat, soll gelöscht werden.

Beispiel-1

AdresseLöschen

```
Library "..\Library\Microsoft ADODB.pcl"

Dim recordset As SmallCOM.ADODB.Recordset
```

```
Dim fields As SmallCOM.ADO.DB.Fields

Dim field As SmallCOM.ADO.DB.Field

recordset = New SmallCOM.ADO.DB.Recordset

recordset.cursorLocation( Enum( SmallCOM.ADO.DB.CursorLocationEnum,
                                adUseClient ) )

recordset.lockType( Enum( SmallCOM.ADO.DB.LockTypeEnum,
                          adLockBatchOptimistic ) )

recordset.open( "select * from Adresse", activeConnection:
               "Driver={SQL Server}; Server=DBServer;Uid=sa;Pwd=; Database=pubs" )

While recordset.eof = FALSE

    fields = recordset.fields

    field = fields.item( "Name" )

    If field.value = "Müller"

        Then

            recordset.delete

            recordset.updateBatch

            recordset.close

            Return

        Else

            recordset.moveNext

        End If

    End If

Wend

recordset.close
```

Beispiel-2

AdresseLöschen

```
Library "..\Library\Microsoft ADO.DB.pcl"
```

```
Dim connection As SmallCOM.ADO.DB.Connection
```

```
connection = New SmallCOM.ADO.DB.Connection
```

```
connection.open( "Driver={SQL Server};Server=DBServer;Database=pubs",
                 userID: "sa" )
```

```
connection.beginTrans
```

```
connection.execute("delete from Adresse where name = 'Müller'")  
connection.commitTrans
```