clsConnection

Inhaltsverzeichnis

- 1 bClose
- 2 BeginTransaction
- 3 bExecute
- 4 bExists
- 5 bExistsField
- 6 bExistsTable
- 7 bInsert
- 8 bOpen
- 9 bSave
- 10 bSetNewConnection
- 11 CommitTransaction
- 12 CommandTimeout
- 13 ConnectionTimeout
- 14 ConnectionSyntax
- 15 ConnectionType
- 16 dtGetCurrentTimestamp
- 17 lGetIdentity
- 18 lGetTan
- 19 nPort
- 20 nState
- 21 oDataAdapter
- 22 oGetConnection
- 23 oGetDatabases
- 24 oGetFields
- 25 oGetTables
- 26 oGetTransaction
- 27 oGetViews
- 28 oOpenDataReader
- 29 oOpenDataTable
- 30 RollbackTransaction
- 31 sBenutzer
- 32 sDatenbank
- 33 SetConnection
- 34 sGetCurrentUser
- 35 sServername
- 36 Tag
- 37 vntGetAggregate
- 38 vntLookup

Die Aufgaben-Center Klasse clsConnection

Mit Hilfe der Klasse **clsConnection** können Verbindungen zu einer Datenbank aufgebaut werden, die Klasse stellt Methoden zum Datenbankzugriff zur Verfügung.
Die Klasse verwendet eine integrierte Protokollierung um Datenbankzugriffe und auftretende Fehler festzuhalten, die Protokollierung wird über die Klasse clsDebug realisiert.

1 bClose

Die Funktion **bClose** schließt eine aktive Verbindung zur Datenbank, ist die Aktion erfolgreich liefert die Funktion den Wert *True* ansonsten *False* zurück.

```
bClose() As Boolean
```

2 BeginTransaction

Die Funktion **BeginTransaction** erzeugt eine neue Transaktion auf der aktiven Datenverbindung, die Transaktion kann über die Funktionen **CommitTransaction** oder **RollbackTransaction** abgeschlossen werden.
Hinweis: Es kann nur eine aktive Transaktion pro Datenverbindung erzeugt werden, parallele Transaktionen werden nicht unterstützt.

```
BeginTransaction()
```

3 bExecute

Die Funktion **bExecute** sendet eine SQL-Anweisung an die Datenbank und liefert bei erfolgreicher Ausführung *True* als Wert zurück.

```
bExecute(sQry As String, Optional sHerkunft As String = "", Optional bErrorLog As Boolean = True, Optional oParameter As clsDataParameter = Nothing) As Boolean
```

Mit dem Argument *sQry* wird die Auszuführende SQL Anweisung angegeben. Mit dem optionalen Argument *sHerkunft* kann angegeben werden wo die Abfrage durchgeführt wird, diese Information wird bei der Protokollierung herangezogen. Das Argument *bErrorLog* bestimmt ob Fehler protokolliert werden sollen. Die Werte für festgelegte Platzhalter innerhalb der SQL Anweisung können durch das optionale Argument *oParameter* mit Hilfe einer ParameterCollection vom Typ clsDataParameter übergeben werden.

Code

```
Dim                    bValid                    As                    Boolean
Dim                    sQry                    As                    String

sQry = "UPDATE KHKArtikel SET Hersteller='LogiSoft' WHERE Artikelnummer='AC-2010-51' AND Mandant='" & goInfo.Mandant
bValid                 =                 goInfo.Datenverbindung.bExecute(sQry)

If                              bValid                              Then
                      MsgBox     "Die     Änderungen     wurden     erfolgreich     ges
End If
```

Alles anzeigen

4 bExists

Die Funktion **bExists** prüft ob in einer Tabelle abhängig von einem Filterkriterium Datensätze vorhanden sind.

bExists(sTabelle As String, sFilter As String, Optional sHerkunft As String = "", Optional bErrorLog As Boolean = True, Optional oParameter As clsDataParameter = Nothing) As Boolean

Mit dem Argument *sTabelle* wird der Name der Tabelle übergeben, die überprüft werden soll. Über das Argument *sFilter* wird das Filterkriterium festgelegt. Mit dem optionalen Argument *sHerkunft* kann angegeben werden wo die Abfrage durchgeführt wird, diese Information wird bei der Protokollierung herangezogen. Das Argument *bErrorLog* bestimmt ob Fehler protokolliert werden sollen. Die Werte für festgelegte Platzhalter innerhalb der SQL Anweisung können durch das optionale Argument *oParameter* mit Hilfe einer ParameterCollection vom Typ clsDataParameter übergeben werden.

Code

```
Dim                    bFound                    As                    Boolean
bFound = goInfo.Datenverbindung.bExists("KHKArtikel", "Artikelnummer='AC-2010-51' AND Mandant='" & goInfo.Mandant)

If                    Not                    bFound                    Then
     MsgBox "Der Artikel mit der Artikelnummer 'AC-2010-51' ist nicht vorhanden."
End If
```

5 bExistsField

Die Funktion **bExistsField** prüft ob in einer Tabelle eine bestimmte Spalte vorhanden ist.

bExistsField(sTabelle As String, sFeld As String, Optional sHerkunft As String = "") As Boolean

Mit dem Argument *sTabelle* wird der Name der Tabelle übergeben, in der nach der Spalte gesucht werden soll. Über das Argument *sFeld* wird die zu suchende Spalte festgelegt. Mit dem optionalen Argument *sHerkunft* kann angegeben werden wo die Abfrage durchgeführt wird, diese Information wird bei der Protokollierung herangezogen.

Code

```
Dim bFound As Boolean
bFound = goInfo.Datenverbindung.bExistsField("KHKArtikel", "USER_Status")

If Not bFound Then
    MsgBox "Das Benutzerdefinierte Feld 'Status' ist nicht vorhanden."
End If
```

6 bExistsTable

Die Funktion **bExistsTable** prüft ob eine bestimmte Tabelle vorhanden ist.

```
bExistsTable(sTabelle As String, Optional sHerkunft As String = "") As Boolean
```

Mit dem Argument *sTabelle* wird der Name der Tabelle übergeben, nach der gesucht werden soll. Mit dem optionalen Argument *sHerkunft* kann angegeben werden wo die Abfrage durchgeführt wird, diese Information wird bei der Protokollierung herangezogen.

Code

```
Dim bFound As Boolean
bFound = goInfo.Datenverbindung.bExistsTable("KHKArtikel")

If Not bFound Then
    MsgBox "Die Tabelle 'KHKArtikel' ist nicht vorh
End If
```

7 bInsert

Die Funktion **bInsert** legt einen neuen Datensatz in einer Tabelle mit Hilfe einer ParameterCollection vom Typ clsDataParameter an. Ist die Aktion erfolgreich liefert die Funktion den Wert *True* ansonsten *False* zurück.

```
bInsert(sTablename As String, oParameter As clsDataParameter, Optional bErrorLog As Boolean = True,
Optional sHerkunft As String = "") As Boolean
```

8 bOpen

Die Funktion **bOpen** stellt eine aktive Verbindung zur Datenbank her, ist die Aktion erfolgreich liefert die Funktion den Wert *True* ansonsten *False* zurück.

```
bOpen() As Boolean
```

9 bSave

Die Funktion **bSave** speichert die Werte einer ParameterCollection vom Typ clsDataParameter in die angegebene Tabelle unter Berücksichtigung der Bedingung. Ist die Aktion erfolgreich liefert die Funktion den Wert *True* ansonsten *False* zurück.

```
bSave(sTablename As String, sWhereClause As String, oParameter As clsDataParameter, Optional
bErrorLog As Boolean = True, Optional sHerkunft As String = "") As Boolean
```

10 bSetNewConnection

Die Funktion **bSetNewConnection** stellt mit den angegebenen Parametern eine neue Verbindung zur Datenbank her, ist die Aktion erfolgreich liefert die Funktion den Wert *True* ansonsten *False* zurück.

```
bSetNewConnection(sServer As String, sDatabase As String, sUser As String, sPassword As String,
Optional nPort As Integer = 0, Optional sConnectionStr As String = "") As Boolean
```

Mit Argument *sServer* wird der Servername angeben, mit *sDatabase* die Datenbank an die sich angemeldet werden soll. Das Argument *sUser* für den Benutzernamen und *sPassword* für das Passwort dienen zur Authentifizierung am Datenbankserver. Über *nPort* kann der Netzwerkport zum Datenbankserver bestimmt werden. Alternativ zu diesen Angaben kann über das optionale Argument *sConnectionStr* auch ein kompletter ConnectionString verwendet werden.

11 CommitTransaction

Die Funktion **CommitTransaction** bestätigt eine erfolgreich durchgeführte Transaktion auf der Datenverbindung und beendet die Transaktion die zuvor mit **BeginTransaction** erzeugt wurde. Bei fehlerhaften Transaktionen können diese mit **RollbackTransaction** rückgängig machen.

CommitTransaction()

12 CommandTimeout

Die Eigenschaft **CommandTimeout** gibt die Zeitdauer in Sekunden an, die für die Ausführung einer SQL Abfrage gewartet werden soll.

CommandTimeout As Integer

13 ConnectionTimeout

Die Eigenschaft **ConnectionTimeout** gibt die Zeitdauer in Sekunden an, die beim Verbindungsaufbau zum Datenbankserver gewartet werden soll.

ConnectionTimeout As Integer

14 ConnectionSyntax

Die Eigenschaft **ConnectionSyntax** gibt an, mit welcher Abfragesprache die Datenverbindung arbeitet. Die Eigenschaft liefert einen Wert aus der Aufzählung *clsConnectionTools.eConnectionSyntax*

ConnectionSyntax = clsConnectionTools.eConnectionSyntax

clsConnectionTools.eConnectionSyntax	Wert	Beschreibung
None	-1	Verbindung mit unbekannter Syntax
MSSQLSyntax	1	Verbindung mit Microsoft SQL Syntax
MSAccessSyntax	2	Verbindung mit Microsoft Access Syntax
MySQLSyntax	3	Verbindung mit MySQL Syntax
MSSQLCESyntax	4	Verbindung mit Microsoft SQL CE Syntax
OracleSyntax	5	Verbindung mit Oracle Syntax

PostgreSQLSyntax	6	Verbindung mit PostgreSQL Syntax
InformixSyntax	7	Verbindung mit Informix Syntax
FirebirdSyntax	8	Verbindung mit Firebird Syntax
DB2Syntax	9	Verbindung mit IBM DB2 Syntax
SQLiteSyntax	10	Verbindung mit SQLite Syntax
SageNCSyntax	11	Verbindung mit Sage New Classic ODBC Syntax
SQL92Syntax	0	Verbindung mit ANSI-SQL 92 Syntax

15 ConnectionType

Mit der Eigenschaft **ConnectionType** kann festgelegt werden zu welchem Datenbanksystem eine Verbindung hergestellt werden soll, bevor diese mit der Funktion **bSetNewConnection** aufgebaut wird.

ConnectionType = clsInfoDatenbanken.eDatenverbindung

clsInfoDatenbanken.eDatenverbindung	**Wert**	**Beschreibung**
eDatenSQL	0	MS-SQL Datenbank
eDatenODBC	1	ODBC Datenverbindung
eDatenOLEDB	2	OLE-DB Datenverbindung
eDatenMySQL	3	MySQL Datenbank
eDatenSageOL	4	Sage Office Line Datenbank
eDatenSageNC	5	Sage New Classic Datenbank

Die Eigenschaft **ConnectionType** kann unter .NET bereits über den Konstruktur der clsConnection Klasse bestimmt werden.

Code

```
Dim                 oConnection                 As                 clsConnection
Dim                     bValid                    As                      Boolean

oConnection  =  New  clsConnection(clsInfoDatenbanken.eDatenverbindung.eDatenSQL)
bValid   =   oConnection.bSetNewConnection("localhost",   "acdata",   "sa",   "")

If                                  bValid                                  Then

End If
```

Alles anzeigen

16 dtGetCurrentTimestamp

Die Funktion **dtGetCurrentTimestamp** liefert einen aktuellen Zeitstempel des Datenbankservers.

dtGetCurrentTimestamp() As Date

17 lGetIdentity

Die Funktion **lGetIdentity** gibt für den zuletzt angelegten Datensatz die ID als Integer zurück. Dies Funktioniert nur für die Tabellen bei denen eine ID-Spalte als Autowert deklariert ist.

lGetIdentity() As Integer

18 lGetTan

lGetTan(sTabelle As String, Optional nMandant As Integer = 0, Optional sZiel As String = "KHKTan") As Integer

19 nPort

Über die Eigenschaft **nPort** kann die Netzwerk Port-Nummer des Datenbankservers ermittelt bzw. gesetzt werden.

```
nPort As Integer
```

20 nState

Über die Eigenschaft **nState** kann der Zustand der Datenverbindung abgefragt werden. Das Ergebnis entspricht der ConnectionState Auflistung aus dem System.Data Namespace.

```
nState As Integer
```

21 oDataAdapter

```
oDataAdapter(sQry As String, Optional sHerkunft As String = "", Optional bErrorLog As Boolean = True,
Optional oParameter As clsDataParameter = Nothing) As DbDataAdapter
```

22 oGetConnection

Die Funktion **oGetConnection** liefert das zugrunde liegende ADO.NET Connection-Objekt zurück.

```
oGetConnection() As DbConnection
```

23 oGetDatabases

Die Funktion **oGetDatabases** ermittelt die vorhandenen Datenbanken zur Datenverbindung. Und gibt diese über einen Binärbaum vom Typ clsBinaryTree mit den String-Elementen zurück.

```
oGetDatabases(Optional sHerkunft As String = "") As clsBinaryTree
```

24 oGetFields

Die Funktion **oGetFields** ermittelt die Felder zu der angegebenen Tabelle *sTabelle*. Und gibt diese über einen Binärbaum vom Typ clsBinaryTree mit den Elementen vom Typ clsDataField zurück.

```
oGetFields(sTabelle As String, Optional sHerkunft As String = "") As clsBinaryTree
```

25 oGetTables

Die Funktion **oGetTables** ermittelt die vorhandenen Tabellen zur verbundenen Datenbank. Und gibt diese über einen Binärbaum vom Typ clsBinaryTree mit den String-Elementen zurück.

```
oGetTables(Optional sHerkunft As String = "") As clsBinaryTree
```

26 oGetTransaction

Die Funktion **oGetTransaction** liefert innerhalb einer Transaktion das zugrunde liegende ADO.NET Transaction-Objekt zurück.

```
oGetTransaction() As DbTransaction
```

27 oGetViews

Die Funktion **oGetViews** ermittelt die vorhandenen Views zur verbundenen Datenbank. Und gibt diese über einen Binärbaum vom Typ clsBinaryTree mit den String-Elementen zurück.

```
oGetViews(Optional sHerkunft As String = "") As clsBinaryTree
```

28 oOpenDataReader

```
oOpenDataReader(sQry As String, Optional sHerkunft As String = "", Optional bErrorLog As Boolean =
True, Optional oParameter As clsDataParameter = Nothing) As clsDataReader
```

29 oOpenDataTable

```
oOpenDataTable(sQry As String, Optional sHerkunft As String = "", Optional bErrorLog As Boolean = True,
Optional oParameter As clsDataParameter = Nothing) As DataTable
```

30 RollbackTransaction

Die Funktion **RollbackTransaction** setzt eine Transaktion die zuvor mit der Funktion **BeginTransaction** begonnen wurde zurück.
Hierdurch werden alle Datenänderungen seit dem Beginn der Transaktion rückgängig gemacht.

```
RollbackTransaction()
```

31 sBenutzer

Die Eigenschaft **sBenutzer** gibt den Namen des Benutzers der mit der Datenbank verbunden ist zurück.

```
sBenutzer As String
```

32 sDatenbank

Die Eigenschaft **sDatenbank** gibt den Namen der aktuell verbundenen Datenbank zurück.

```
sDatenbank As String
```

33 SetConnection

Über die **SetConnection** Funktion kann man eine vorhandene ADO.NET Datenverbindung übergeben.

Danach kann man mit dem Objekt der *clsConnection* Klasse auf der Datenverbindung arbeiten.

```
SetConnection(oConnection As DbConnection)
```

34 sGetCurrentUser

Die Funktion **sGetCurrentUser** gibt den aktuell angemeldeten Benutzer der Datenverbindung zurück.

```
sGetCurrentUser() As String
```

35 sServername

Die Eigenschaft **sServername** gibt den Namen des aktuell verbundenen Datenbankservers zurück.

```
sServername As String
```

36 Tag

Über die **Tag** Eigenschaft kann man ein beliebiges Objekt an die Datenverbindung hängen.

```
Tag() As Object
```

37 vntGetAggregate

```
vntGetAggregate(sTabelle As String, sFilter As String, Optional sFeld As String = "", Optional
sAggregateFunction As String = "", Optional sHerkunft As String = "", Optional bErrorLog As Boolean =
True, Optional oParameter As clsDataParameter = Nothing)As Object
```

38 vntLookup

vntLookup(sField As String, sTable As String, sClause As String, Optional vntDefault As Object = Nothing, Optional sHerkunft As String = "", Optional bErrorLog As Boolean = True, Optional oParameter As clsDataParameter = Nothing) As Object