22.4.3 Klasse DB (gb.db)

Diese Klasse repräsentiert die aktuelle Datenbank-Verbindung, die standardmäßig die zur ersten geöffneten Datenbank ist. Sie können die Datenbank-Verbindung ändern, indem Sie den Wert der Eigenschaft `DB.Current` modifizieren.

22.4.3.1 Eigenschaften

Die Klasse *DB* verfügt über die folgenden Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Charset	String	Gibt den von der Datenbank verwendeten Zeichensatz zurück.
Current	Connection	Setzt die aktuelle DB-Verbindung oder gibt die aktuelle DB-Verbindung zurück.
Debug	Boolean	Legt mit True fest, dass die Datenbank-Komponente in den Debugging-Modus versetzt wird. Wenn dieses Flag gesetzt ist, so wird jede Datenbank-Abfrage auf der Standard-Fehlerausgabe ausgegeben.
Error	Integer	Gibt den Fehler-Code des letzten, vom Datenbank-Treiber ausgelösten Fehler zurück.
Handle	Pointer	Gibt den Zeiger auf die Standard-Datenbank-Verbindung zurück. Dies ist ein Zeiger auf das zugrundeliegende Objekt - assoziiert mit der Standard-Datenbank-Verbindung.
IgnoreCharset	Boolean	Gibt True zurück, wenn der Datenbank-Zeichensatz ignoriert werden soll oder setzt den Wert der Eigenschaft.
Opened	Boolean	Gibt True zurück, wenn die aktuelle DB-Verbindung geöffnet ist.
Databases	.Connection.Databases	Gibt eine Übersicht aller Datenbanken zurück, die vom DB-Server verwaltet werden.
Users	.Connection.Users	Gibt eine Übersicht der DB-Benutzer in einer Datenbank zurück.
Tables	.Connection.Tables	Gibt eine Übersicht der verwalteten Tabellen in einer Datenbank zurück.

Tabelle 22.4.3.1.1 : Eigenschaften der Klasse DB

Die Klasse .Connection.Databases repräsentiert eine Sammlung aller Datenbanken, die vom Datenbank-Server verwaltet werden. Möglicherweise sehen Sie nicht jede Datenbank, wenn der Benutzer, mit dem Sie sich mit dem Server verbunden haben, nicht über ausreichende Rechte verfügt.

- Die Eigenschaft .Connection.Databases.Count vom Datentyp Integer gibt die Anzahl Datenbanken auf dem DB-Server an.
- Die Methode .Connection.Databases.Add (Name As String) legt eine neue Datenbank mit dem Datenbanknamen 'Name' an.
- Die boolsche Funktion .Connection.Databases.Exist (Key As String) ergibt True, wenn die angegebene Datenbank in der Übersicht der verwalteten Datenbanken existiert.
- Mit der Methode .Connection.Databases.Remove (T_Name As String) wird die angegebene Datenbank 'T_Name' gelöscht.
- Die Methode .Connection.Databases.Refresh ( ) erneuert die Übersicht der verwalteten Datenbanken. Der interne Cache wird dabei gelöscht.

Die Klasse .Connection.Tables stellt eine Übersicht aller Tabellen der Datenbank bereit, zu der Sie sich mit dem Verbindungsobjekt verbunden haben.

- Diese Klasse erbt von .SubCollection.
- Diese Klasse ist virtuell.
- Diese Klasse ist nicht erzeugbar.
- Diese Klasse verhält sich wie ein schreibgeschütztes Array.
- Diese Klasse ist mit dem Schlüsselwort FOR EACH aufzählbar.

- Die Eigenschaft .Connection.Tables.Count (Typ Integer) gibt die Anzahl Tabellen in der Übersicht der verwalteten Tabellen in einer Datenbank an.
- Die Methode .Connection.Tables.Add (Name As String [, Type As String]) legt eine neue Tabelle mit dem Tabellenname 'Name' an. Der (optionale) Parameter 'Type' legt den Tabellen-Typ fest und ist nur sinnvoll, wenn ein MySQL-Datenbank-Server verwendet wird. Folgende Typen werden unterstützt: InnoDB, BDB, HEAP, ISAM, MERGE und MYISAM. Beachten Sie: Nur InnoDB- und Berkley-DB-Tabellen erlauben die Nutzung von Transaktionen!
- Die boolsche Funktion .Connection.Tables.Exist (Key As String) ergibt True, wenn die angegebene Tabelle in der Übersicht der verwalteten Tabellen in einer Datenbank existiert.
- Mit der Methode .Connection.Tables.Remove (T_Name As String) wird die angegebene DB-Tabelle 'T_Name' in der Datenbank gelöscht.
- Die Methode .Connection.Tables.Refresh ( ) erneuert die Übersicht der verwalteten Tabellen in einer Datenbank. Der interne Cache wird dabei gelöscht.

Die Klasse .Connection.Users liefert eine Übersicht aller registrierten DB-Benutzer auf dem DB-Server. SQLite-Datenbanken jedoch haben keine speziellen DB-Benutzer; sie akzeptieren jeden Benutzer, weil ein ausgewiesenes Sicherheitskonzept fehlt.

- Die Eigenschaft .Connection.Users.Count (Datentyp Integer) gibt die Anzahl DB-User zurück, die vom DB-Server verwaltet werden.
- Mit der Methode .Connection.Users.Add(Name As String [, Password As String, Admin As Boolean]) wird ein neuer Benutzer auf dem Datenbankserver registriert. *Name* ist der Name des DB-Benutzers. Passwort ist ein optionales Passwort. Setzen Sie den Parameter 'Admin' auf True, wenn Sie wollen, dass dieser DB-Benutzer ein DB-Administrator sein soll. Ein Datenbank-Administrator hat das Recht, Datenbanken und Benutzer zu erzeugen oder zu entfernen und kann sich mit jeder Datenbank auf dem Server verbinden.
- Die Function .Connection.Users.Exist (Key As String) (Datentyp Boolean) ergibt True, wenn der angegebene Benutzer in der Übersicht aller registrierten DB-Benutzer auf dem DB-Server existiert.
- Mit .Connection.Users.Refresh ( ) wird die Übersicht aller registrierten DB-Benutzer auf dem DB-Server erneuert. Der interne Cache wird dabei gelöscht.
- Die Methode .Connection.Users.Remove (Name As String) löscht einen DB-User vom DB-Server.

22.4.3.2 Methoden

Die Klasse *DB* verfügt über diese Methoden:

Methode	Beschreibung
Begin ( )	Startet eine Transaktion.
Close ( )	Schließt eine bestehende DB-Verbindung zwischen DB-Client und DB-Server.
Open ( )	Öffnet eine (neue) Verbindung zu der in den Verbindungseigenschaften angegebenen Datenbank. Die Open()-Methode besitzt keine Parameter. Deshalb sind die Verbindungseigenschaften vor dem Aufruf Open()-Methode einzustellen.
Commit ( )	Änderungen am Datenbestand werden übernommen.
Rollback ( )	Änderungen am Datenbestand während einer Sitzung werden zurückgenommen.
Delete (Table As String [, Request As String, Arguments As , ...])	Löscht Datensätze aus einer DB-Tabelle (Umsetzung einer SQL-Anweisung…). *Table* ist der Name der DB-Tabelle. *Request* ist eine SQL-WHERE-Klausel zum Filtern der Tabelle und *Arguments* werden gemäß der SQL-Syntax in Anführungszeichen gesetzt und innerhalb der Abfragezeichenfolge ersetzt.
Edit (Table As String [, Request As String, Arguments As , ...]) As Result	Gibt ein Lese/Schreib-Ergebnis-Objekt für das Editieren von Datensätzen in der angegebenen Tabelle zurück. *Table* ist der Name der DB-Tabelle. *Request* ist eine SQL-WHERE-Klausel zum Filtern der Tabelle und *Arguments* werden gemäß der SQL-Syntax in Anführungszeichen gesetzt und innerhalb der Abfragezeichenfolge ersetzt.
Exec (Request As String, Arguments As , ...) As Result	Führt eine beliebige SQL-Anfrage aus und gibt das Ergebnis der Anfrage als ein Nur-Lese-Ergebnis zurück. *Request* ist jede valide SQL-Anweisung und *Arguments* werden gemäß der SQL-Syntax in Anführungszeichen gesetzt und innerhalb der Abfragezeichenfolge ersetzt.

Methode	Beschreibung
Find (Table As String [, Request As String, Arguments As , ...]) As Result	Gibt ein schreibgeschütztes Ergebnisobjekt zurück, das zur Abfrage von Datensätzen in der angegebenen Tabelle verwendet wird. Table ist der Name der DB-Tabelle. *Request* ist eine SQL-WHERE-Klausel zum Filtern der Tabelle und *Arguments* werden gemäß der SQL-Syntax in Anführungszeichen gesetzt und innerhalb der Abfragezeichenfolge ersetzt.
Function Limit (Limit As Integer) As Connection	Begrenzt die Anzahl der Datensätze, die von der nächsten Abfrage zurückgegeben werden. Nachdem die Abfrage ausgeführt wurde, wird das Limit automatisch aufgehoben. Der Funktionswert ist vom Typ Connection, so dass Sie etwa Folgendes schreiben können: DB.Limit(12).Exec(…).
FormatBlob (Data As String) As String	Formatiert einige Blob-Daten so, dass sie in eine SQL-Abfrage eingefügt werden können.
Quote (Name As String [, Table As Boolean]) As String	Gibt einen Bezeichner in Anführungszeichen zurück, so dass Sie ihn frei in eine Abfrage einfügen können. Dieser Bezeichner kann ein Tabellen- oder ein Feldname sein.
Function Subst (Format As String, Arguments As , ...) As String	Erzeugt einen SQL-Satz, indem seine Argumente in eine Format-Zeichenkette eingesetzt werden. *Format* ist die SQL-Anweisung und *Arguments* ist die Liste der zu ersetzenden Argumente.

Tabelle 22.4.3.2.1 : Methoden der Klasse DB

Beispiel

```
DIM $hDBConnecton As NEW Connection

WITH $hDBConnecton
   .Type = "postgresql"
   .Host = "localhost"
   .Login = "loginname"
   .Password = "password"
   .Name = "testdb"
END WITH

TRY $hDBConnecton.Open()
IF Error THEN PRINT "Cannot Open Database! Error = "; Error.Text
```

Hinweise:

- Mit der Methode Delete(…), können Sie SQL-Anweisungen schreiben, die unabhängig vom zugrunde liegenden Datenbanktyp sind. Ein nicht zu unterschätzender Vorteil, wenn es um die Entwicklung und den Test wiederverwendbarer Software geht.
- Sobald Sie das Ergebnis-Objekt erhalten haben, können Sie einige der Felder ändern. Danach können Sie die Result.Update()-Methode aufrufen, um die Änderungen an die Datenbank zu senden.

```
    DIM hResult AS Result
    DIM sCriteria AS String
    DIM iParemeter AS Integer

    sCriteria = "id = &1"
    iParameter = 1012

    $hDBConnecton.Begin()

'-- Same as "SELECT * FROM table_name WHERE id = 1012"
    hResult = $hDBConnecton.Edit("table_name", sCriteria, iParameter)
'-- Set field value
    hResult!Name = "Mayer"

'-- Update the value
    hResult.Update()
    $$hDBConnecton.Commit()
```

- Das Quoting ist abhängig vom Datenbankserver-Treiber, so dass diese Methode verwendet werden sollte, wenn Sie datenbankunabhängigen Code schreiben müssen:

```
'-- Gibt die Anzahl der Datensätze in einer Abfrage zurück. sTable ist der Name der Tabelle.
'-- Er kann reservierte Zeichen enthalten, daher müssen Sie ihn in Anführungszeichen setzen!
    rResult = Handle.Exec("SELECT COUNT(*) AS nRecord FROM " & DB.Quote(sTable, True))
```

```
    PRINT rResult!nRecord
```

- Die Subst()-Funktion können Sie so verwenden: Die &1, &2...-Muster im Format-String werden durch die SQL-Darstellung der 1., 2.... Argumente ersetzt. Diese Argumente werden gemäß der zugrunde liegenden Datenbank-SQL-Syntax angegeben.

```
PRINT DB.Subst("WHERE Name = &1 AND Date = &2", "Mayer-Motzen", Now())
```