22.4.1 Klasse Connection (gb.db)

Die Klasse Connection ist Bestandteil der Komponente gb.db und repräsentiert eine Verbindung zu einer Datenbank. Eine Instanz kann durch Quelltext erzeugt werden.

Achtung: Die Klassen DB (ebenfalls Teil der Komponente gb.db) und Connection verfügen über diese gleichen Eigenschaften:

```
Charset Databases Error IgnoreCharset Opened Tables Users Version
```

und über jene gleichen Methoden:

```
Begin Close Commit Create Delete Edit Exec Find FormatBlob Limit Open Quote Rollback Subst
```

Die gemeinsamen Eigenschaften und Methoden werden im Kapitel 22.4.3 beschrieben.

Dieses Kapitel widmet sich den speziellen Eigenschaften der Klasse Connection:

```
Host Login Name Password Port Timeout Type User
```

22.4.1.1 Eigenschaften

Die Klasse *Connection* verfügt über die speziellen Eigenschaften:

Eigenschaft	Datentyp	Beschreibung
Host	String	Liefert oder setzt den Host, auf dem sich der Datenbank-Server befindet. Dieser Host kann ein Rechnername oder eine IP-Adresse sein. Der Standardhost ist localhost. Bei SQLite ist der Hostname der absolute Pfad für die Datenbank mit dieser Eigenschaft.
Login	String	Liefert oder setzt den DB-Benutzer, der für den Verbindungsaufbau verwendet wird.
User	String	Synonym für die Eigenschaft Login.
Name	String	Liefert oder setzt den Namen der Datenbank, mit der DB-Benutzer sich verbinden soll. Wenn Sie keinen Namen angeben, dann wird eine Standard-System-Datenbank verwendet.
Password	String	Ermittelt das Password, das für den Verbindungsaufbau benötigt wird oder legt das Passwort fest.
Port	String	Liefert oder setzt den TCP/IP-Port, der für den Verbindungsaufbau verwendet wird.
Timeout	Integer	Liefert oder setzt den Timeout für den Verbindungsaufbau in Sekunden. Der Standard-Timeout beträgt 20 Sekunden.
Type	String	Repräsentiert den Typ des Datenbank-Servers oder der DB-Bibliothek bei SQLite, mit dem Sie eine Datenbank-Verbindung herstellen möchten.

Tabelle 22.4.1.1.1 : Eigenschaften der Klasse Connection

Hinweise:

- Der *Host* kann ein Rechnername oder eine IP-Adresse sein. Der Standard-Host ist 'localhost'. Für SQLite ist der Host-Name der Pfad für die Datenbank-Datei. Bei MySQL/MariaDB kann der Hostname den Pfad zu einer Socket-Datei enthalten. Dann muss die Zeichenkette mit einem Schrägstrich "/" beginnen. Für ODBC ist der Hostname die DSN-Namensdefinition. Informationen zu ODBC finden Sie im Kapitel → 22.9.0 ODBC (Open Database Connectivity).
- Wenn Sie keinen Datenbank-Namen angeben, wird eine Standard-System-Datenbank verwendet. Für PostgreSQL ist die Standard-Datenbank template1, für MySQL ist es mysql und für SQLite ist es /tmp/sqlite.db. Für SQLite ist es eine In-Memory-Datenbank /tmp/sqlite.db. Für SQLite können Sie auch eine Datenbank im Speicher erzeugen, indem als Datenbankname ':memory:' angegeben wird. Selbstverständlich benötigt der MySQL- oder PostgreSQL-Benutzer mindestens Lese- und Verbindungszugriff auf diese Datenbanken.
- Der Typ des Datenbank-Servers muss in Kleinbuchstaben angegeben werden! Die aktuell unterstützten Datenbank-Server sind: PostgreSQL: "postgresql", MySQL: "mysql", SQLite 2: "sqlite2", SQLite 3: "sqlite3", SQLite 2 or SQLite 3: "sqlite" und ODBC: "odbc".

- Der Standard-Port hängt vom Verbindungstyp ab. Für MySQL ist es '3306' und für PostgreSQL '5432'. Bei SQLite wird die Eigenschaft *Port* nicht verwendet.
- Für SQLite ist das Password leer, weil SQLite kein DB-Benutzer-Konzept kennt. Der Zugriff auf eine SQLite-Datenbank wird durch die Dateiberechtigungen der Datenbankdatei gesteuert. Der SQLite-DB-User ist immer der System-User, der das DB-Programm aktuell ausführt.

22.4.1.2 Erzeugen einer DB-Verbindung

Mit der folgenden Prozedur können Sie bei einem Programm-Start prüfen, ob der DB-Server MySQL oder PostgreSQL gestartet wurde, zu dem eine DB-Verbindung hergestellt werden soll:

```
'' Determining the status of the DB server<br>
'' Argument: DBMS types mysql or postgresql
Public Sub DBServerStatus(argType As String)

    Dim sMessage As String

    If Not argType Or (Lower(argType) <> "mysql" And Lower(argType) <> "postgresql") Then
       sMessage = "<b><font size='+1', color='DarkRed'>"
       sMessage &= ("Error!")
       sMessage &= "</b></font><hr>"
       sMessage &= ("The DB server name is missing or has been written incorrectly.")
       sMessage &= gb.NewLine & gb.NewLine
       sMessage &= "<b>" & ("The application is therefore terminated!") & "</b>"
       Message.Error(sMessage)
   '-- The (main) program is terminated. ------------------------------------------------
       Quit
    Endif

    Exec ["systemctl", "status", argType] Wait For Read Write

'-- EXIT STATUS IN PROCESS.LASTVALUE
'-------------------------------------------------------------
'-- Documented in `man systemctl`
'-- 0 -> program is running or service is OK | unit is active
'-- 3 -> program is not running | unit is not active
'-- 4 -> program or service status is unknown | no such unit

    Select Case Process.LastValue
      Case 4
        sMessage = "<b><font size='+1', color='DarkRed'>"
        sMessage &= ("Error!")
        sMessage &= "</b></font><hr>"
        sMessage &= Subst("&1 &2 &3", ("The DB server"), Upper(argType), ("is obviously not installed!"))
        sMessage &= gb.NewLine & gb.NewLine
        sMessage &= "<b>" & ("The application is therefore terminated!") & "</b>"
        Message.Error(sMessage)
    '-- The (main) program is terminated. ----------------------------------------
        Quit
      Case 3
        sMessage = "<b><font size='+1', color='DarkRed'>"
        sMessage &= ("Error!")
        sMessage &= "</b></font><hr>"
        sMessage &= Subst("&1 &2 &3", ("The DB server"), Upper(argType), ("is not started!"))
        sMessage &= gb.NewLine & gb.NewLine
        sMessage &= "<b>" & ("The application is therefore terminated!") & "</b>"
        Message.Error(sMessage)
    '-- The (main) program is terminated. ----------------------------------------
        Quit
    End Select

End
```

Um eine Verbindung zu einer Datenbank herzustellen, erzeugen Sie zuerst ein Verbindungsobjekt, setzen dann die benötigten Eigenschaften und rufen anschließend die Methode 'Open' auf. Im folgenden Abschnitt werden Ihnen unterschiedliche Möglichkeiten vorgestellt, wie Sie eine DB-Verbindung erzeugen. Es wird vorausgesetzt, dass für das verwendete DBMS ein DB-Benutzer (Name, Passwort) und eine Datenbank existieren. Die Datenbank kann auch leer sein – also ohne DB-Tabellen.

```
Dim hConnection As Connection
hConnection = New Connection ( [ DatabaseURL As String ] )
```

Das Format des optionalen Parameters *DatabaseURL* ist

```
$TYPE://[$USER@]$HOST[:$PORT]/$DBNAME
```

wobei die Teile $USER und $PORT optional sind. Wenn dieser Parameter 'DatabaseURL' jedoch angegeben und gültig ist, so werden die Verbindungseigenschaften auf die übergebenen Eigenschaftswerte gesetzt.

(1) MySQL-DB-Verbindung (DB-Benutzer 'test', Passwort 'test', Port 3306, Datenbank-Name 'test')

```
Dim hConnection As Connection

hConnection = New Connection("mysql://test@localhost:3306/test")
hConnection.Password = "test"

Try hConnection.Open()
If Error Then Print "The database cannot be opened! Error = "; Error.Text
```

Alternative:

```
Dim hConnection As Connection

WITH hConnection
 .Type        = "mysql"
 .Host        = "localhost"
 .User        = "test"
 .Password    = "test"
 .Name        = "test"
END WITH

Try hConnection.Open()
If Error Then Print "The database cannot be opened! Error = "; Error.Text
```

(2) SQLite3-DB-Verbindung (Datenbank-Name 'test', Pfad 'User.Home/Test')

```
Dim hConnection As Connection

hConnection = New Connection("sqlite3://" & User.Home /& "Test" &/ "test.sqlite")

Try hConnection.Open()
If Error Then Print "The database cannot be opened! Error = "; Error.Text
```

Wenn kein DB-Name angegeben wurde, so wird eine DB-Verbindung zu einer temporären Datenbank erzeugt. Alternativ können Sie auch diese Anweisung verwenden:

```
Dim hConnection As Connection

hConnection = New Connection("sqlite3://:memory/")
```

(3) PostgreSQL-DB-Verbindung (Benutzer 'test', Passwort 'test', Port 5432, Datenbank-Name 'test')

```
Dim hConnection As Connection

hConnection = New Connection("postgresql://test@localhost:5432/test")
hConnection.Password = "test"

Try hConnection.Open()
If Error Then Print "The database cannot be opened! Error = "; Error.Text
```

Alternative:

```
Dim hConnection As Connection

WITH hConnection
 .Type        = "postgresql"
 .Host        = "localhost"
 .Login       = "test"
 .Password    = "test"
 .Name        = "test"
END WITH

Try hConnection.Open()
If Error Then Print "The database cannot be opened! Error = "; Error.Text
```

22.4.1.3 Erzeugen einer DB-Verbindung 1 – Modul oder statische Klasse

Sie können das Erzeugen einer DB-Verbindung in ein Modul oder in eine statische Klasse auslagern.

So wird das Modul oder eine statische Klasse zu einem Singleton. Es stellt sicher, dass für jede Datenbank in Ihrem Programm stets nur eine DB-Verbindung zu einer bestimmten Datenbank existiert. Beim ersten Zugriff auf die DB wird die DB-Verbindung aufgebaut – danach bleibt diese zur Laufzeit des Programms bestehen! Das stellt sicher, dass Ihr Programm immer *diese eine DB-Verbindung* verwendet und keine gleichzeitigen Verbindungen zur selben Datenbank hält, was zu Datenbank-Deadlocks führen kann. Es wäre ja bei Multitask-Systemen durchaus möglich, zugleich mehrere DB-Verbindungen zu einer Datenbank offen zu halten. Dann kann es aber zu Situationen kommen, bei der in der einen DB-Verbindung auf etwas gewartet wird, was in der anderen DB-Verbindung nicht freigegeben wird, weil dort auf etwas gewartet wird, was in der anderen DB-Verbindung erst freigegeben werden muss.

Das ist der Quelltext für die statische Klasse DBCS, die stets in allen Beispiel-Projekten im Buch verwendet wird:

```
[1]  ' Gambas class file
[2]
[3]  Create Static
[4]
[5]  Property Read DBConnection As Connection '-- It is a 'DataBaseConnectionSingleton'
[6]  Private $DBConnection As New Connection
[7]
[8]  Private Function DBConnection_Read() As Connection
[9]
[10]     If Not $DBConnection.Opened Then
[11]      ' ------------------------------------------------------------------------------------
[12]        $DBConnection.Type = "sqlite"
[13]        $DBConnection.Port = Null
[14]        $DBConnection.User = Null
[15]        $DBConnection.Password = Null
[16]        $DBConnection.Host = MCreateDir.DBHost    '-- File-Path for SQLite-Host
[17]        $DBConnection.Name = "flowers.sqlite"     '-- Name of the SQLite database
[18]      ' ------------------------------------------------------------------------------------
[19]      ' $DBConnection.Type = "postgresql"
[20]      ' $DBConnection.Port = 5432
[21]      ' $DBConnection.User = "test"
[22]      ' $DBConnection.Password = "test"
[23]      ' $DBConnection.Host = "localhost"          '-- DBHost is `localhost ` or  an IP adddress
[24]      ' $DBConnection.Name = "test"               '-- Name of the database
[25]      '------------------------------------------------------------------------------------
[26]      ' $DBConnection.Type = "mysql"
[27]      ' $DBConnection.Port = 3306
[28]      ' $DBConnection.User = "test"
[29]      ' $DBConnection.Password = "test"
[30]      ' $DBConnection.Host = "localhost"          '-- DBHost is `localhost ` or an IP adddress
[31]      ' $DBConnection.Name = "test"               '-- Name of the database
[32]      '------------------------------------------------------------------------------------
[33]
[34]        If $DBConnection.Type <> "sqlite" Then
[35]           If $DBConnection.User = Null Or If $DBConnection.Password = Null Then
[36]              Error.Raise(("No database credentials"))
[37]            Endif
[38]        Endif
[39]        $DBConnection.Open()
[40]     Endif
[41]
[42]     Return $DBConnection
[43]
[44]     Catch
[45]     Error.Propagate()
[46]
[47] End
```

Beispiel für die Verwendung einer in der o.a. Klasse DBCS definierten DB-Verbindung:

```
Public Function GetReportData() As Result

'-- Returns the result of a database query as a database result.
    sSQLStatement = "SELECT vorname,nachname,wohnort FROM contats ORDER BY nachname"

    hDBResult = DBCS.DBConnection.Exec(sSQLStatement)

    Return hDBResult

End
```

22.4.1.4 Erzeugen einer DB-Verbindung 2 – IDE

Die Gambas-IDE bietet die u.a. Möglichkeit, eine DB-Verbindung (SQLite, MySQL, PostgreSQL, ODBC) für Testzwecke zu definieren und herzustellen. Wenn Sie die Komponente gb.db ausgewählt haben, dann wird automatisch ein neues Verzeichnis 'Verbindungen' im Projekt-Verzeichnis erzeugt. Über dessen Kontext-Menü können Sie in der IDE eine neue DB-Verbindung erzeugen oder eine bestehende DB-Verbindung aktualisieren:

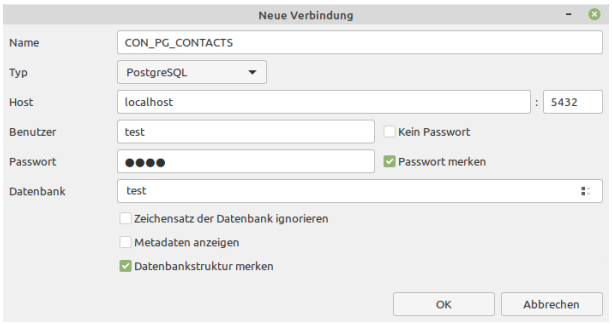

Abbildung 22.4.1.4.1: Neue DB-Verbindung zu einer PostgreSQL-Datenbank 'test' erzeugen

Wenn Sie eine DB-Verbindung in der IDE öffnen, dann können Sie

- sich Informationen zu den Feldern der ausgewählten DB-Tabelle ansehen,
- sich zusätzlich System-Tabellen anzeigen lassen,
- neue DB-Tabellen anlegen,
- alle Eingaben speichern,
- alle DB-Daten neu laden,
- eine DB-Tabelle löschen,
- eine DB-Tabelle umbenennen,
- eine DB-Tabelle kopieren,
- eine DB-Tabelle einfügen oder
- eine Text-Datei importieren (Dialog → Default-Dateityp: *.csv).

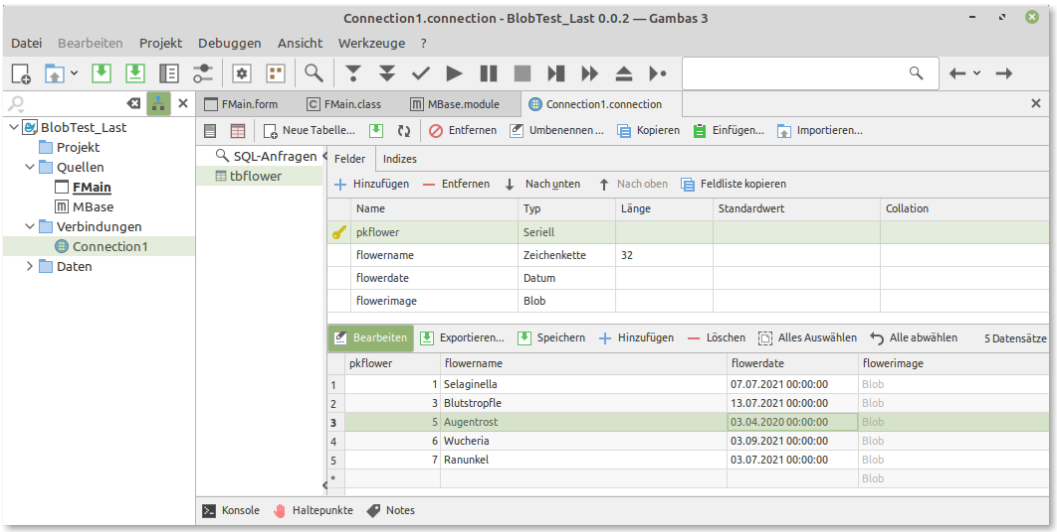

Abbildung 22.4.1.4.2: Übersicht zu DB-Feldern und Indizes einer DB-Tabelle (SQLite3)

Sie können aber auch DB-Daten einer ausgewählten DB-Tabelle bearbeiten. Voraussetzung ist jedoch, dass Sie in den Bearbeiten-Modus (ON/OFF-Schalter) umschalten.

Sie können

- Daten in einer DB-Tabelle bearbeiten (einfügen, ändern, löschen),
- Daten einer DB-Tabelle in eine csv-Datei exportieren,
- vorgenommene Änderungen von Daten speichern,

- einen Datensatz hinzufügen,
- einen oder mehrere Datensätze löschen,
- alle Datensätze auswählen oder
- alle Datensätze de-markieren.

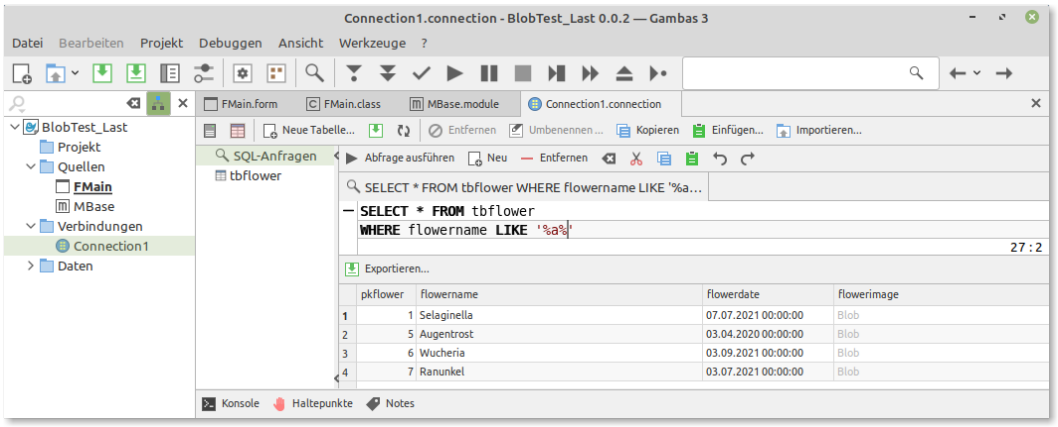

Abbildung 22.4.1.4.3: SQL-Anfrage für eine SQLite-DB-Tabelle → Connection1

Die Möglichkeit, auch SQL-Anfragen für eine ausgewählte DB-Tabelle zu stellen, rundet den Umfang der Arbeit mit DB-Tabellen für eine bestimmte DB-Verbindung in der IDE ab.