Beginner Tutorial: Using SQLite with BlitzMax - Part 1

** If you have enjoyed these tutorials, you may want to check out my other contributions (link)

Table of Contents

Introduction

SQLite is a relational database management system contained in a relatively small C library. It is a public domain project created by D. Richard Hipp. For more info head to their website (www.sqlite.org).

Due to its simple design, many people have enabled the use of the SQLite library using various programming languages including Blitzmax. Two wrappers for BlitzMax are available, one written by Noel Cower and the other by TeaMonkey.

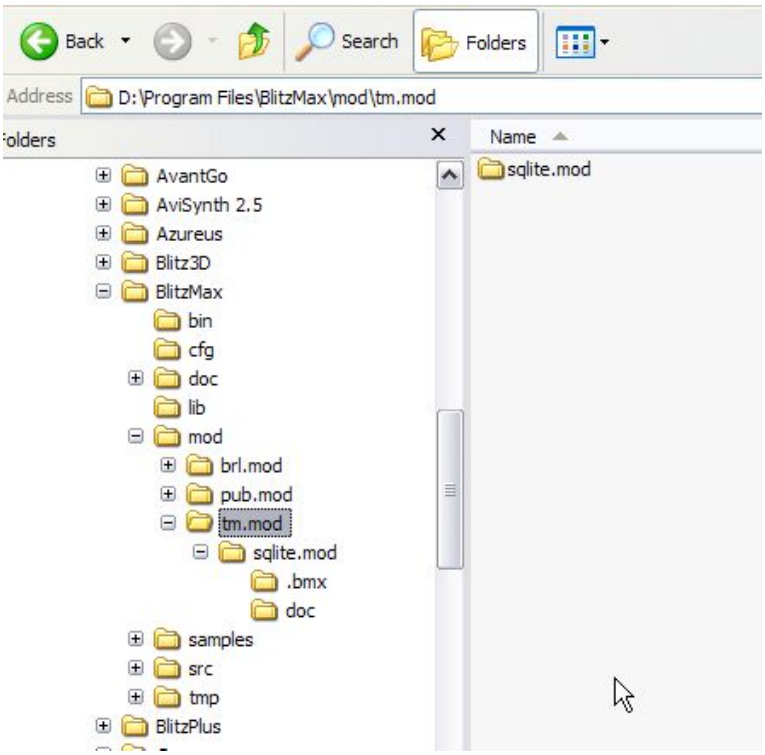
For this tutorial, I will be using the module written by TeaMonkey.

Basically the only thing required to use SQLite from within BlitzMax is to copy the TeaMonkey Module into the proper location in the BlitzMax Mod folders and some knowledge of SQLite.

Downloading the SQLite Module

The first thing to do is to download the module. I have recompiled the TeaMonkey module for BlitzMax 1.18 as the version on TeaMonkey's download site does not work with version 1.18.

Download the module from here: tm.zip (~1.5MB) and then extract into your BlitzMax mod folder. After the extraction your mod should have a tm.mod folder (see pic below)



Now we are ready to start writing our first SQLite program using BlitzMax. Note that users of

Mac OS and Linux will have to do a MakeMods before using the module.

Writing the first SQLite BlitzMax Program

Cut and paste the program below into the MaxIDE and run it.

```
SuperStrict
Import tm.SQLite

Local db:Int 'Database Handle
Local Command:String

SQLite3_Open(":memory:", db)

Command="CREATE TABLE Player(Name TEXT, Ranking INTEGER)"
SQLite3_Exec(db, Command, Null, Null, Null)

Command="INSERT INTO Player VALUES('Ronaldinho', 1)"
SQLite3_Exec(db, Command,Null, Null, Null)

Command="SELECT * FROM Player"
SQLite3_Exec(db, Command, CallBack, Null, Null)

SQLite3_Close(db)
End

Function CallBack:Int(user_data:Byte Ptr, num_cols:Int, value:Byte Ptr Ptr,
column_name:Byte Ptr Ptr)

   For Local i:Int=0 Until num_cols
      Print String.FromCString(Column_name[i])+": "+String.FromCString(value[i])
   Next

   Return 0
End Function
```

You should see on the Output screen, two lines of Output from the SQL query (see below)

```
Building untitled2
Compiling:untitled2.bmx
flat assembler  version 1.64
3 passes, 4944 bytes.
Linking:untitled2.debug.exe
Executing:untitled2.debug.exe
Name: Ronaldinho
Ranking: 1

Process complete
```

Understanding our First SQLite Program

It is highly recommended to use either **Strict** or **Superstrict** so that BlitzMax will catch any typos

```
SuperStrict
```

The **Import** statement will allow us to use the SQLite module in our program

```
Import tm.SQLite
```

Initialise the variables we are going to need in our program

```
Local db:Int 'Database Handle
Local Command:String
```

The **SQLite3_Open** function opens a database file for us to play with. The integer variable **db** returns the handle of the database. In this particular example, the database will be held in memory.

```
SQLite3_Open(":memory:", db)
```

The **SQLite3_Exec** function allows us to send SQL commands to the database engine. We are creating a new table called **Player** with fields called **Name** and **Ranking** of type Text and Integer respectively using the **SQL CREATE TABLE** command.

```
Command="CREATE TABLE Player(Name TEXT, Ranking INTEGER)"
SQLite3_Exec(db, Command, Null, Null, Null)
```

Just to clarify, there is sometimes a confusion between a database file and database tables. A Database File is like a container which holds the tables. Data are kept in the database tables.

Once we have the **Player** table defined, we can now start to insert values into the table. In this instance we are inserting the player named **Ronaldinho** with a number **1** ranking. This is done via the SQL command **INSERT INTO table-name VALUES(x,y).**

```
Command="INSERT INTO Player VALUES('Ronaldinho', 1)"
SQLite3_Exec(db, Command,Null, Null, Null)
```

Note that text are kept within apostrophes whereas BlitzMax uses quotes.

Now that we have insert data into our table, we can start to query the database. This is done via the SQL command **SELECT * FROM table-name**.

```
Command="SELECT * FROM Player"
SQLite3_Exec(db, Command, CallBack, Null, Null)
```

Notice now that we have added another parameter (**CallBack**) into the function rather then **Null** as per the previous exec command.

The reason for this is that the SQLite engine, as it retrieves data from the table, will allow the user to 'catch' the data a record at a time via the CallBack function.

The CallBack function must be of the following form.

```
Function CallBack:Int(user_data:Byte Ptr, num_cols:Int, value:Byte Ptr Ptr,
column_name:Byte Ptr Ptr)
```

For the moment, what is of interest to know is that the number of columns (or fields) returned by the SELECT query will be returned in the **num_cols** integer variable. The field name will be in the **column_name** variable and the value of the field in the **value** variable.

The CallBack should always return a zero unless you want to indicate an error then return a non-zero.

```
    Return 0
End Function
```

You can then put within the CallBack function the statements you need to capture the returned records. In this particular example I have chosen to simply print the Fieldname plus its associated value.

```
Function CallBack:Int(user_data:Byte Ptr, num_cols:Int, value_name:Byte Ptr Ptr,
column_name:Byte Ptr Ptr)

    For Local i:Int=0 Until num_cols
        Print String.FromCString(Column_name[i])+": "+String.FromCString(value_name[i])
    Next

    Return 0
End Function
```

The last thing to do then is to close our database and free up whatever memory has been allocated before we end.

```
SQLite3_Close(db)
End
```

A slighty more complicated example

The example above is the simplest I can think of to show you the principles involved in using SQLite from within BlitzMax. In the next tutorial we will see a slightly more complicated example showing a bit more of the capaibility of using SQLite.

On to Part 2.