The Dirty Dozen: Connecting to Nearly Any Database with the MBS SQL Plugin

Christian Schmitz
CEO
Monkeybread Software

MBS SQL Plugin

- Alternative interface to databases for Xojo (Real Studio)
- Two interfaces:
  - SQLDatabaseMBS
    - based on Xojo's database class
  - SQLConnectionMBS
    - our native interface for C++ library
- can be mixed.

SQLDatabaseMBS

- Based on Xojo's database class
- Using RecordSet for query results
- Using Prepared Statements
- Using DatabaseRecord class for insert
- No Edit/Update/Delete on RecordSet

SQLConnectionMBS

- Native Interface
- SQLCommandMBS for prepared statement & results
- SQLAPIMBS classes for API specific functions
- SQLite Backup
- SQLite Encryption functions

Supported databases

- Centura SQLBase
- DB2
- Firebird
- Informix
- InterBase
- MariaDB
- Microsoft Access
- Microsoft SQL Server
- MySQL
- ODBC
- Oracle Database Server
- PostgreSQL
- SQL Anywhere
- SQLite
- SQLCipher
- Sybase

Features

- Bring your own native library
- MySQL / MariaDB
  - Commercial, Open Source or Embedded Library
- SQLite
  - SQL Cipher Library
  - SpatiaLite Library
  - Built-in SQLite inside plug-in
  - Preinstalled SQLite on Mac/Linux

Prepared Statements

- Parameters by ID, by Index or by Name
  - p.Bind("FirstName") = Value
- With type or auto detect type
  - :1, :name or ?
- Bind all via Dictionary

Prepared Statements

```
// by index
dim r as PreparedSQLStatement = db.Prepare(
    "Insert into test_tbl(fid, fvarchar20) values(:1, :2)")

r.BindType(0, SQLPreparedStatementMBS.kTypeLong)
r.BindType(1, SQLPreparedStatementMBS.kTypeString)

r.Bind(0, 12345)
r.Bind(1, "Hello World by index")

r.SQLExecute
```

Prepared Statements

```
// by name
dim sql as string =
      "Insert into test_tbl(fid, fvarchar20) values(:fid, :fvarchar20)"
dim v as Variant = db.Prepare(sql)
dim p as SQLPreparedStatementMBS = v

p.BindType("fid", SQLPreparedStatementMBS.kTypeLong)
p.BindType("fvarchar20", SQLPreparedStatementMBS.kTypeString)

p.Bind("fid", 2345)
p.Bind("fvarchar20", "Hello World by name")

p.SQLExecute
```

Multithreading

- Perform SQL Execute & SQL Select multithreaded
- Connect multithreaded
- In Xojo thread, keep GUI running

BLOB Handling

- Set/Get BLOB as Memoryblock or String
- Stream data from/to file or stream
- DataConsumerMBS
  - get chunks of data in event
- DataProviderMBS
  - provide data in chunks via event

RecordSet movements

- Scrollable RecordSets
- Optionally local caching of RecordSet
- Move First, Next, Previous & Last

Debugging

- LastStatment property
- Trace Events
- See Parameters and Fields in Debugger

Exceptions

- SQLDatabaseMBS
  - Error Property
  - Exceptions **off** by default
- SQLConnectionMBS
  - Error Property
  - Exceptions **on** by default
- Exceptions are better to make sure you get all errors!

SQLite

- Internal SQLite library
- Supports AES128, AES256 and RC4 encryption.
- Updated regularly to latest SQLite +SEE version.
- SetBusyTimeout and SetBusyHandler
- Added FTS5 and JSON extensions.

PostgreSQL

- Internal PostgreSQL library available for Mac/Linux
- Use notifications to inform other clients:
  - PostgresNotification event
  - Listen method

Events

- Trace event
  - Log all SQL commands
- DidConnect & WillConnect
  - Log connections
  - Set Parameters for connection
- Working
  - Called while multithreading
  - Allows cancellation

Example Connect

```
dim con as new SQLConnectionMBS

// where is the library?
con.Option(con.kOptionLibrarySQLite) = "/usr/lib/libsqlite3.0.dylib"

// connect to database
con.Connect("test.db", "", "", SQLConnectionMBS.kSQLiteClient)

MsgBox "We are connected!"

Exception error as SQLErrorExceptionMBS
    MsgBox error.message
```

Example Insert

```
dim cmd as new SQLCommandMBS(con)

// Insert row
cmd.setCommandText("Insert into TestTable(id, Name) values(:1, :2)")

// bind by ID
cmd.Param(1).setAsLong(2)
cmd.Param(2).setAsString("Bob")

// Insert first row
cmd.Execute
```

Example Insert

```
dim cmd as new SQLCommandMBS(con)

// Insert row
cmd.setCommandText( _
    "Insert into TestTable(id, Name) values(:id, :Name)")

// bind by Name
cmd.Param("id").setAsLong(2)
cmd.Param("Name").setAsString("Bob")

// Insert first row
cmd.Execute
```

Example Query

```
// create command object
dim cmd as new SQLCommandMBS(con, "Select id, Name from TestTable")

// Select from our test table
cmd.Execute

// fetch results row by row and print results
while cmd.FetchNext

    dim fid as integer = cmd.Field("id").asLong
    dim fName as string = cmd.Field("Name").asStringValue

    window1.Listbox1.AddRow str(fid), fName
wend
```

Example Connect

```
dim db as new SQLDatabaseMBS

// where is the library?
con.Option(con.kOptionLibrarySQLite) = "/usr/lib/libsqlite3.0.dylib"

// connect to database
db.DatabaseName = "sqlite:" + "test.db"

if db.Connect then
    MsgBox "We are connected!"
end if
```

Example Insert

```
dim d as new DatabaseRecord

d.IntegerColumn("id")=2
d.Column("someText")="test insert"
d.BlobColumn("blob")="Just a test"

db.InsertRecord("TestTable", d)

if db.Error then
    MsgBox db.ErrorMessage
end if
```

Example Query

```
dim r as RecordSet = db.SQLSelect("Select id, Name from TestTable")

// fetch results row by row and print results
while not r.EOF
    dim fid as integer = r.Field("id").IntegerValue
    dim fName as string = r.Field("Name").StringValue

    window1.Listbox1.AddRow str(fid), fName

    r.MoveNext
wend
```

Intermix Interfaces

- Query SQLConnectionMBS for SQLDatabaseMBS
  - to use native APIs
- Convert SQLCommandMBS to RecordSet
  - e.g. for Reports
- Run SQLSelect on SQLConnectionMBS for RecordSet

Other Features

- Bulk Row Fetch
- Microsoft SQL from Mac/Linux via FreeTDS
- Run Stored Procedures
- use SQLite Library with ICU
- AutoCommit on/off

SQLite Shell

```
dim arguments() as string

dim f as FolderItem = SpecialFolder.Desktop.Child("test.sqlite")
dim o as FolderItem = SpecialFolder.Desktop.Child("out.csv")

arguments.Append "sqlite3"   // path to app
arguments.Append f.NativePath
arguments.Append "-csv"      // CSV export
arguments.Append "-header" // Include header line
arguments.Append "-cmd"      // Output file
arguments.Append ".output "+o.NativePath
arguments.Append "-cmd"      // Query
arguments.Append "select * from Documentation;"

dim n as integer = InternalSQLiteLibraryMBS.Shell(arguments)
```

MBS SQL Plugin

- Free to Try
  - Over 80 Examples
- Cost: $149 USD

- Platforms
  - Desktop, Console & Web
  - Mac OS X, Windows, Linux, not for iOS
  - 64-bit, 32-bit and ARM

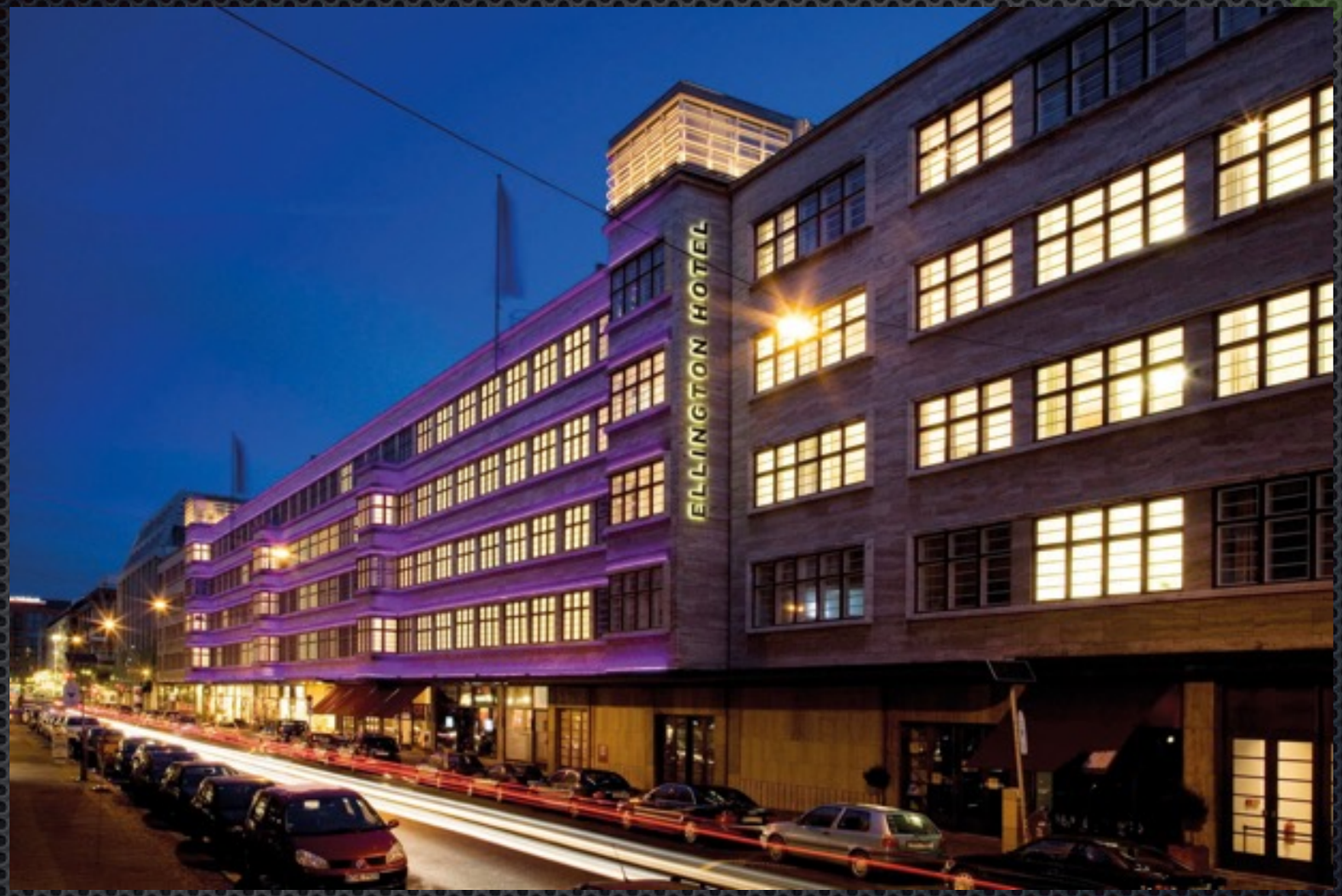
Q & A

Christian Schmitz

support@monkeybreadsoftware.de

Give us feedback on this session in the XDC app!

See you in Berlin!



4/5th May 2017