BIJU@JUST4SHARING.COM (MAILTO:BIJU@JUST4SHARING.COM) +919741600911

Welcome Guest [Log In (../../Account/Login.aspx)]

Using SQLite with Visual Studio 2010

Recently I had a chance to work with SQLite to be used as dynamic storage media for Logging messages from Microsoft Message Queue. As I was new to SQLite, did a bit of research on it and found it bit easy and interesting. Hope my writing will be of some help to you. I will cover the below points.

1. What is System.Data.SQLite?
2. How to integrate System.Data.SQLite with your project as a package
3. How to open a connection
4. How to create a table dynamically
5. How to insert a record
6. How to check if a table exists or not

1. What is System.Data.SQLite

System.Data.SQLite is the original SQLite database engine and a complete ADO.NET 2.0/3.5 provider all rolled into **a single mixed mode** assembly. It is a **complete drop-in replacement** for the original sqlite3.dll . There is no need to have sqlite3.dll in your development box at all.

I was little confused in the beginning. I downloaded all three at first.

· sqlite-dll-win32-x64-3071300.zip (1.17 MiB)
· sqlite-shell-win32-x86-3071300.zip(262.59 KiB)
· sqlite-netFx35-setup-bundle-x64-2008-1.0.81.0.exe (6.79 MiB)

latter I found that third exe alone will serve my purpose. There is an easy way of integrating with Visual Studio without using any of this exes. I will explain below.

2. How to integrate System.Data.SQLite with your project as a package

· Download the precompiled binary package for your target framework and processor architecture (e.g. 32-bit x86, .NET Framework 2.0).
· Extract the package to a directory named "Externals" inside your project directory.
· Add a reference to the "System.Data.SQLite" assembly from the "Externals" directory.
· If necessary (i.e. you require LINQ support), also add a reference to the "System.Data.SQLite.Linq" assembly from the "Externals" directory.
· Alternatively, when using Visual Studio 2010, you can simply use the NuGet package that corresponds to your target processor architecture. Installing the assemblies into the Global Assembly Cache is not recommended as it may cause conflicts with other applications installed on the machine.

3. How to open a new connection to SQlite Database.

```
string inputFile = "test.s3db"
string dbConnection = String.Format("Data Source={0}", inputFile)
SQLiteConnection cnn = new SQLiteConnection(dbConnection);
cnn.Open();
```

Note: *If database file does not exist then new file will be created with the above command. It is not necessary that there should be db file before opening connecton.*

4. How to create a table dynamically using SQLite command

```
// Define the SQL Create table statement
    string createLogTableSQL = "CREATE TABLE [Log] (" +
        "[ID] INTEGER PRIMARY KEY AUTOINCREMENT," +
        "[Message] TEXT  NULL" +
        ")";
    using (SQLiteTransaction sqlTransaction = cnn.BeginTransaction())
    {
        // Create the table
        SQLiteCommand createCommand = new SQLiteCommand(createLogTableSQL, cnn);
        createCommand.ExecuteNonQuery();
        createCommand.Dispose();
        // Commit the changes into the database
        sqlTransaction.Commit();
    } // end using
```

5. How to insert a record in to a table

```
string sql  = "insert into Log (Message) values ('This is sample      data');"
SQLiteConnection cnn = new SQLiteConnection(dbConnection);
cnn.Open();
SQLiteCommand mycommand = new SQLiteCommand(cnn);
mycommand.CommandText = sql;
int rowsUpdated = mycommand.ExecuteNonQuery();
cnn.Close();
```

6. How to check if a table exists or not

```
using (SQLiteConnection cnn = new SQLiteConnection(dbConnection))
{
               cnn.Open();
               String sql = string.Format("SELECT COUNT(*) FROM sqlite_master
                  WHERE type = 'table'                 AND name ='{0}'", tableName);
               count = ExecuteScalar(sql);
}
```

7. Create your helper class to manage crud operations with SQLite

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SQLite;
using System.IO;
using System.Configuration;


    public class SQLiteDatabase
    {
        protected String dbConnection;

        /// <summary>
        ///     Single Param Constructor for specifying the DB file.
        /// </summary>
        /// <param name="inputFile">The File containing the DB</param>
        public SQLiteDatabase(string DBDirectoryInfo, String inputFile)
        {
            string sourceFile = Path.Combine(DBDirectoryInfo, inputFile);
            dbConnection = String.Format("Data Source={0}", sourceFile);
        }

        /// <summary>
        ///     Single Param Constructor for specifying advanced connection options.
        /// </summary>
        /// <param name="connectionOpts">A dictionary containing all desired options and their
               values</param>
        public SQLiteDatabase(Dictionary<String, String> connectionOpts)
        {
            String str = "";
            foreach (KeyValuePair<String, String> row in connectionOpts)
            {
                str += String.Format("{0}={1}; ", row.Key, row.Value);
            }
            str = str.Trim().Substring(0, str.Length - 1);
            dbConnection = str;
        }

        /// <summary>
        ///     Allows the programmer to run a query against the Database.
        /// </summary>
        /// <param name="sql">The SQL to run</param>
        /// <returns>A DataTable containing the result set.</returns>
        public DataTable GetDataTable(string sql)
        {
            DataTable dt = new DataTable();
            try
            {
                SQLiteConnection cnn = new SQLiteConnection(dbConnection);
                cnn.Open();
                SQLiteCommand mycommand = new SQLiteCommand(cnn);
                mycommand.CommandText = sql;
                SQLiteDataReader reader = mycommand.ExecuteReader();
                dt.Load(reader);
                reader.Close();
                cnn.Close();
            }
            catch (Exception ex)
            {
            }
            return dt;
        }

        /// <summary>
        ///     Allows the programmer to interact with the database for purposes other than a quer
        /// </summary>
        /// <param name="sql">The SQL to be run.</param>
        /// <returns>An Integer containing the number of rows updated.</returns>
        public int ExecuteNonQuery(string sql)
        {
            SQLiteConnection cnn = new SQLiteConnection(dbConnection);
            cnn.Open();
            SQLiteCommand mycommand = new SQLiteCommand(cnn);
            mycommand.CommandText = sql;
            int rowsUpdated = mycommand.ExecuteNonQuery();
            cnn.Close();
            return rowsUpdated;
        }

        /// <summary>
        ///     Allows the programmer to retrieve single items from the DB.
        /// </summary>
        /// <param name="sql">The query to run.</param>
        /// <returns>A string.</returns>
        public string ExecuteScalar(string sql)
        {
            SQLiteConnection cnn = new SQLiteConnection(dbConnection);
            cnn.Open();
            SQLiteCommand mycommand = new SQLiteCommand(cnn);
            mycommand.CommandText = sql;
            object value = mycommand.ExecuteScalar();
            cnn.Close();
            if (value != null)
            {
                return value.ToString();
            }
            return "";
        }

        /// <summary>
        ///     Allows the programmer to easily update rows in the DB.
        /// </summary>
        /// <param name="tableName">The table to update.</param>
        /// <param name="data">A dictionary containing Column names and their new values.
</param>
        /// <param name="where">The where clause for the update statement.</param>
        /// <returns>A boolean true or false to signify success or failure.</returns>
        public bool Update(String tableName, Dictionary<String, String> data, String where)
        {
            String vals = "";
            Boolean returnCode = true;
            if (data.Count >= 1)
            {
                foreach (KeyValuePair<String, String> val in data)
                {
                    vals += String.Format(" {0} = '{1}',", val.Key.ToString(), val.Value.ToString(
                }
                vals = vals.Substring(0, vals.Length - 1);
            }
            try
            {
                this.ExecuteNonQuery(String.Format("update {0} set {1} where {2};", tableName,
                                     vals, where));
            }
            catch(Exception ex)
            {
                returnCode = false;
                //ServiceLogWriter.LogError(ex);
            }
            return returnCode;
        }

        /// <summary>
        ///     Allows the programmer to easily delete rows from the DB.
        /// </summary>
        /// <param name="tableName">The table from which to delete.</param>
        /// <param name="where">The where clause for the delete.</param>
        /// <returns>A boolean true or false to signify success or failure.</returns>
        public bool Delete(String tableName, String where)
        {
            Boolean returnCode = true;
            try
            {
                this.ExecuteNonQuery(String.Format("delete from {0} where {1};", tableName, where)
            }
            catch (Exception ex)
            {
                returnCode = false;
            }
            return returnCode;
        }

        /// <summary>
        ///     Allows the programmer to easily insert into the DB
        /// </summary>
        /// <param name="tableName">The table into which we insert the data.</param>
        /// <param name="data">A dictionary containing the column names and data for the insert.
</param>
        /// <returns>A boolean true or false to signify success or failure.</returns>
        public bool Insert(String tableName, Dictionary<String, String> data)
        {
            String columns = "";
            String values = "";
            Boolean returnCode = true;
            foreach (KeyValuePair<String, String> val in data)
            {
                columns += String.Format(" {0},", val.Key.ToString());
                values += String.Format(" '{0}',", val.Value);
            }
            columns = columns.Substring(0, columns.Length - 1);
            values = values.Substring(0, values.Length - 1);
            try
            {
                this.ExecuteNonQuery(String.Format("insert into {0}({1}) values({2});", tableName,
            }
            catch (Exception ex)
            {
                returnCode = false;
            }
            return returnCode;
        }

        /// <summary>
        ///     Allows the programmer to easily delete all data from the DB.
        /// </summary>
        /// <returns>A boolean true or false to signify success or failure.</returns>
        public bool ClearDB()
        {
            DataTable tables;
            try
            {
                tables = this.GetDataTable("select NAME from SQLITE_MASTER where type=
                 'table' order by NAME;");
                foreach (DataRow table in tables.Rows)
                {
                    this.ClearTable(table["NAME"].ToString());
                }
                return true;
            }
            catch
            {
                return false;
            }
        }

        /// <summary>
        ///     Allows the user to easily clear all data from a specific table.
        /// </summary>
        /// <param name="table">The name of the table to clear.</param>
        /// <returns>A boolean true or false to signify success or failure.</returns>
        public bool ClearTable(String table)
        {
            try
            {

                this.ExecuteNonQuery(String.Format("delete from {0};", table));
                return true;
            }
            catch
            {
                return false;
            }
        }

        /// <summary>
        ///     Allows the programmer to easily test connect to the DB.
        /// </summary>
        /// <returns>A boolean true or false to signify success or failure.</returns>
        public bool TestConnection()
        {
            using (SQLiteConnection cnn = new SQLiteConnection(dbConnection))
            {
                try
                {
                    cnn.Open();
                    return true;
                }
                catch
                {
                    return false;
                }
                finally
                {
                    // Close the database connection
                    if ((cnn != null) && (cnn.State != ConnectionState.Open))
                        cnn.Close();
                }
            }
        }

        /// <summary>
        ///     Allows the programmer to easily test if table exists in the DB.
        /// </summary>
        /// <returns>A boolean true or false to signify success or failure.</returns>
        public bool IsTableExists(String tableName)
        {
            string count = "0";
            if (dbConnection == default(string))
                return false;
            using (SQLiteConnection cnn = new SQLiteConnection(dbConnection))
            {
                try
                {
                    cnn.Open();
                    if (tableName == null || cnn.State != ConnectionState.Open)
                    {
                        return false;
                    }
                    String sql = string.Format("SELECT COUNT(*) FROM sqlite_master WHERE type
                = 'table' AND name ='{0}'", tableName);
                    count = ExecuteScalar(sql);
                }
                finally
                {
                    // Close the database connection
                    if ((cnn != null) && (cnn.State != ConnectionState.Open))
                        cnn.Close();
                }
            }
            return Convert.ToInt32(count) > 0;
        }
}
```

Tags: ASP.NET, SQLite

Related Blogs

Live Traffic Feed

A visitor from Canada arrived from google.ca and viewed "How to use SQLite database in Visual Studio" 18 secs ago

A visitor from Makati, Mindoro Occidental arrived from google.com.ph and viewed "How to do a simple RDLC report using Visual Studio" 36 mins ago

A visitor from United States arrived from google.com and viewed "Service Applications and their databases in SharePoint 2013" 57 mins ago

A visitor from Mexico, Distrito Federal arrived from google.com.mx and viewed "How to use SQLite database in Visual Studio" 1 hr 11 mins ago

A visitor from Montgomery, Alabama viewed "Configuring SharePoint 2013 to support workflow Management Service" 1 hr 16 mins ago

A visitor from United States arrived from google.com and viewed "KeyValuePair VS DictionaryEntry" 1 hr 33 mins ago

Real-time view · Get Feedjit

About

Welcome to my blogs. My SharePoint and .NET Experience is shared for everyone. Mail me (mailto:biju@just4sharing.com?Subject=My%20Article) your suggestions and they are always welcome.

EMail Subscriptions

Would love to receive new blog alerts in your inbox?

Email Address

Subscribe

Featured Blogs

How to do a simple RDLC report using Visual Studio (http://tech.just4sharing.com/Pages/ASP/How-to-do-a-simple-RDLC-report-using-Visual-Studio.aspx)

Content Type Hub in SharePoint 2013 (http://tech.just4sharing.com/Pages/tech/Content-Type-Hub-in-SharePoint-2013.aspx)

Configuring Windows Azure Access Control Service and Facebook authentication in SharePoint 2013 – Part 1 (http://tech.just4sharing.com/Pages/tech/Configuring-Windows-Azure-Access-Control-Service-and-Facebook-authentication-in-SharePoint-2013-Part-1.aspx)

Contact Us

Biju Joseph(MCSD)
SharePoint Consultant
Bangalore, INDIA
Phone: +919741600911
Email: biju@just4sharing.com (mailto:biju@just4sharing.com)

Stay Connected