How to use SQLite in Windows Phone

This article explains how to add SQLite support in Windows Phone 8 applications and how to start using it.

28 Apr 2013

Introduction

A very common task for most of the applications is to store and access the data. Almost all applications has to store some kind of user data for a later use. This is the reason why mobile platforms provide ways to the developer handle those data; with Windows Phone 8 is not different, it provides the `System.IO.IsolatedStorage` namespace to access files and/or application settings.

This is normally enough for many applications, but a database would be better if the user wants to store data in a more structured way. This post is about the use of the SQLite database to handle those structured data.

At the time of writing this post, it is not so easy to start using SQLite in a Windows Phone 8 application. So, the first part of this post will give a workaround to do that. The second part will show how to create a connection, a database, tables and how to store and retrieve data. The first part has to be updated in the future, when the startup of SQLite become easier.

Tip: This article covers Windows Phone 8. If you want to develop a database-driven application for Windows Phone 7 devices, then we recommend you use the library sqlitewindowsphone as described in this post. Tests show it works perfectly on Lumia 800!

Adding SQLite support to the Windows Phone 8 application

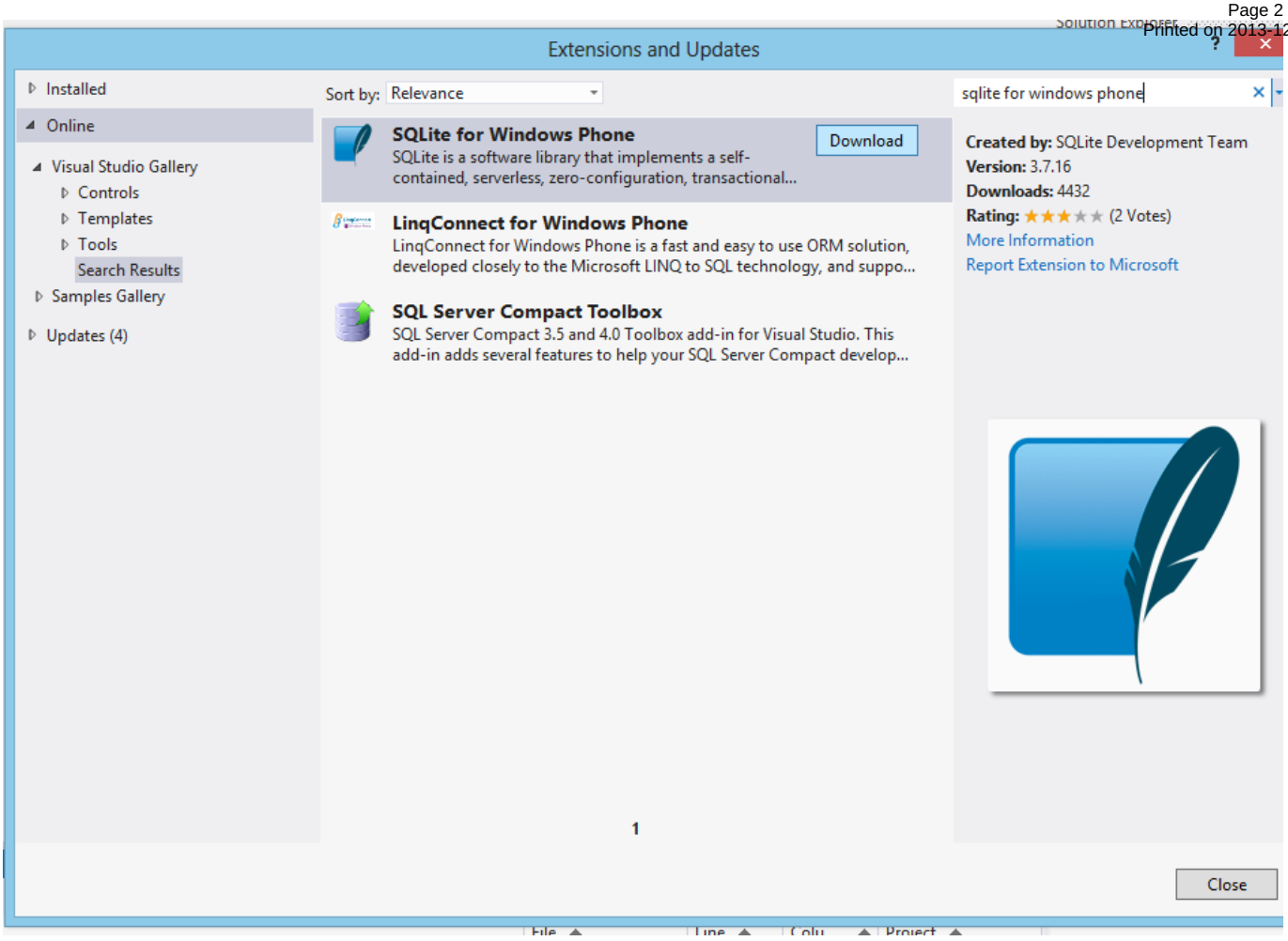
This post will use a simple example to install the SQLite support and handle a database. So, the first thing you have to do is to create an empty Windows Phone application solution. I'll assume the name of the application that you created as "SqliteForWP8".

First of all, I have to say that everything I've done to add SQLite support in "SqliteForWP8" application was based on this post (http://wp.qmatteoq.com/working-with-sqlite-in-windows-phone-8-a-sqlite-net-version-for-mobile/) written by qmatteoq . I'll just filter his post and put here only the needed steps. If you want more details about why these steps are needed, take a look at the post which is mentioned above since there is a lot of useful information there.

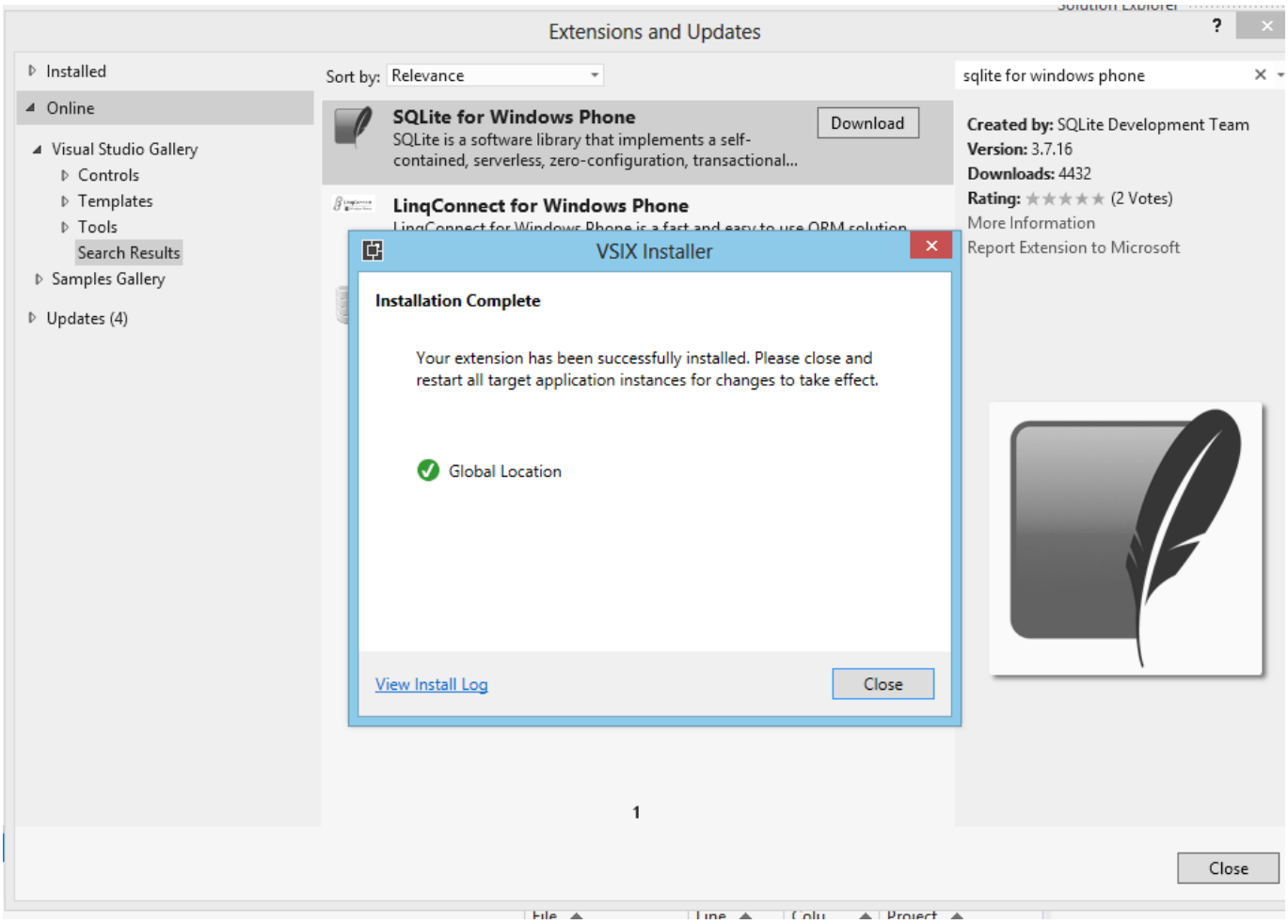
Installing SQLite for Windows Phone SDK

You have to install the **SQLite for Windows Phone SDK** in the Visual Studio 2012 that you are using. To do that, click on TOOLS -> Extensions and Updates -> Online -> Then search for "sqlite for windows phone". It should look like the following picture.

(C) Copyright Nokia 2013. All rights reserved.



Click on Download. You'll have to restart the Visual Studio after the download is completed. The below image is shown when the download and installation is completed.



(C) Copyright Nokia 2013. All rights reserved.

Installing sqlite-net-wp8 wrapper

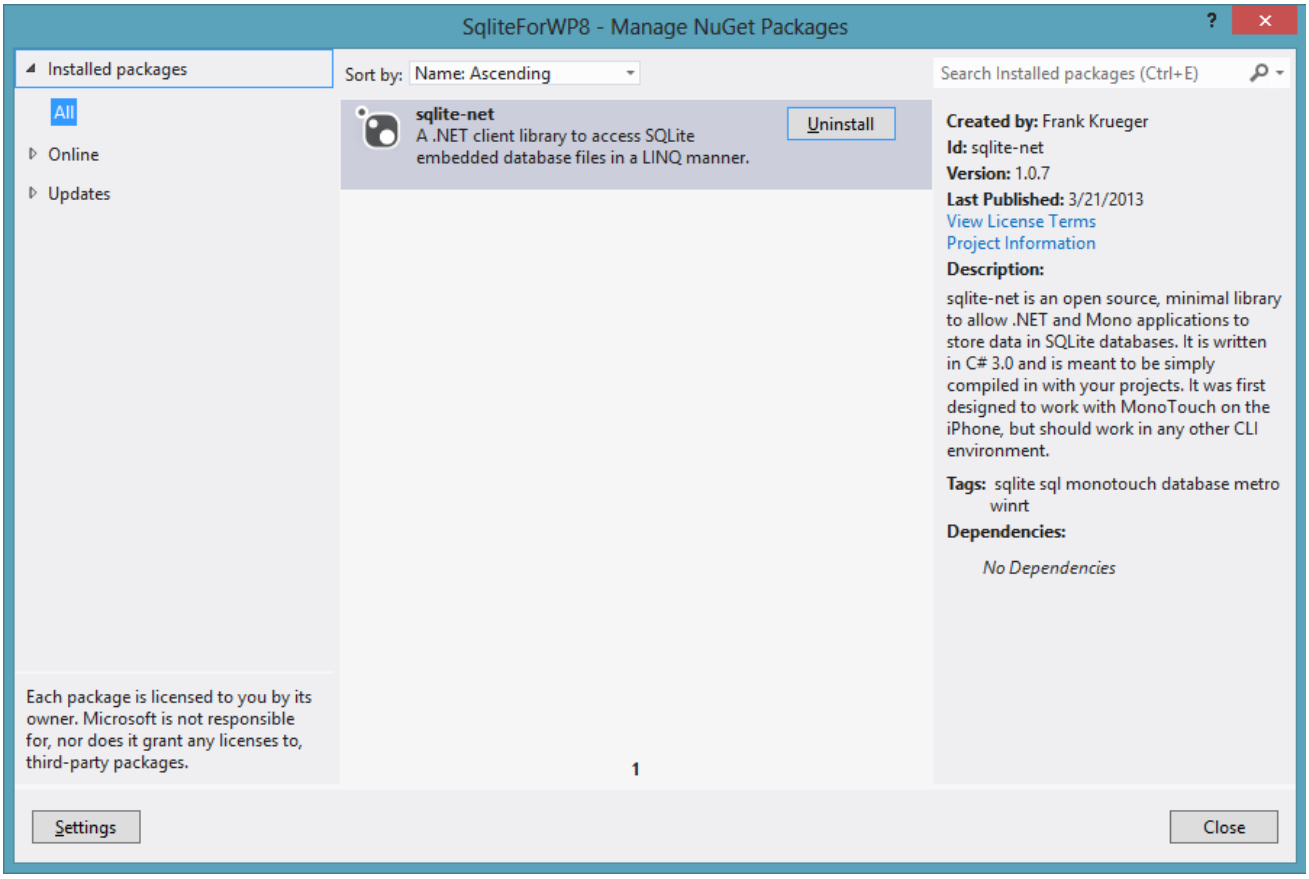
This wrapper is the C++ bridge between the **SQLite for Windows Phone SDK** and the **sqlite-net** NuGet package that we'll install soon. First, you have to clone the following Git repository "https://github.com/peterhuene/sqlite-net-wp8.git". At this point, I'm assuming that you have some knowledge of Git. If you don't know anything about Git, please go to (https://code.google.com/p/tortoisegit/) and install the tortoise Git; it's easy to use and Windows friendly.

Now, you add this project to the "SqliteForWP8" solution that we created before: Right click on the Solution (Not the project) -> Click Add -> Existing Project and select the Sqlite.vcxproj from the folder that you cloned previously using Git. This will create the "Sqlite" C++ project in your solution. Now you add a reference to this project in your "SqliteForWP8" project: Right click on the References folder of "SqliteForWP8" project (Not the solution) -> Click on Add Reference -> Solution and select the "Sqlite" project -> Click on OK button.

If you have done everything right, at this time you have one solution with two projects inside it. The "Sqlite" C++ project and the "SqliteForWP8" C# project.

Installing sqlite-net package

The **sqlite-net** package provides the C# classes that you'll use to handle the sqlite database; storing and retrieving data. First, right click on the "SqliteForWP8" project (Not the Solution) -> Click on "Manage NuGet Packages" and search for "sqlite-net" -> Click on "Install" button. At the time of writing this post, the plugin version was 1.0.7 and it was created by "Frank Krueger". Use the following image as a reference to install the correct one.



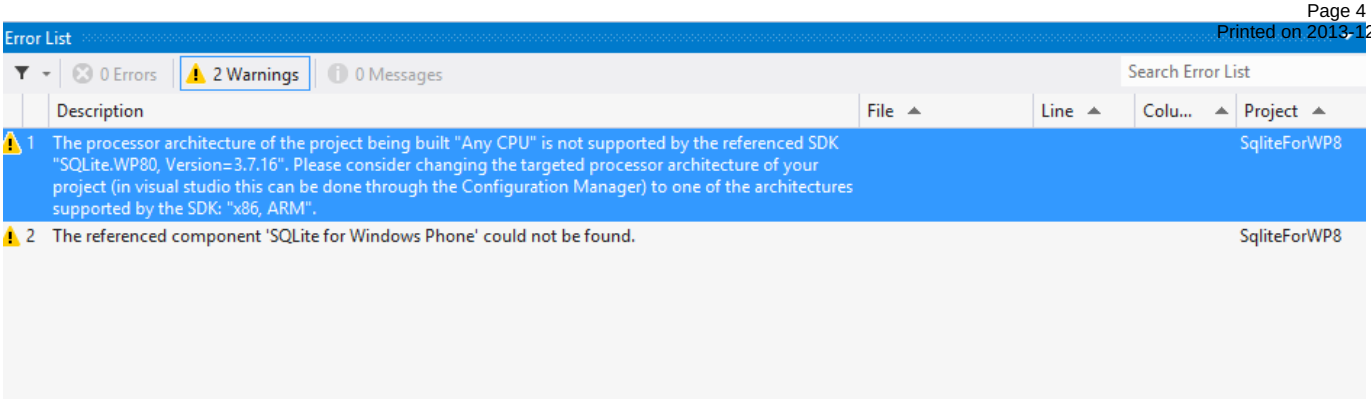
You'll probably get this error message: *The type or namespace name 'Community' could not be found*. It can be fixed by this approach: Create the `USE_WP8_NATIVE_SQLITE` compilation symbol in the "SqliteForWP8" project. Please pay attention to the configuration and platform that you are creating this symbol since each platform has its own set of compilation symbols. This symbol will tell the sqlite-net package that you are using the SQLite for Windows Phone SDK.

Troubleshootings

"Any CPU" problem

If you get a warning message as shown below, follow these steps to fix it: Right click on the Solution -> Click on Configuration Properties -> Configuration Manager and change the active solution platform to x86 (If you are using an emulator) or ARM (If you are using a Windows Phone 8 device).

(C) Copyright Nokia 2013. All rights reserved.



Using the database

Finally, we are able to start using a SQLite database to store/retrieve structured data in our application. To keep the code simple, I implemented everything in the MainPage.xaml.cs file. But in real world, a good practice should be to separate the Data layer from UI layer, as MVVM pattern says.

The `SQLiteConnection` class which is shown in the example code is part of the **sqlite-net** package. It has various methods to control the database but I had shown just a basic usage in this example. To see the complete documentation, go to the https://github.com/praeclarum/sqlite-net/wiki wiki page of the project. If you want to download this example and test it for yourself, click here Media:SQLiteForWP8.zip.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Navigation;
using Microsoft.Phone.Controls;
using Microsoft.Phone.Shell;
using SqliteForWP8.Resources;
using SQLite;
using Windows.Storage;
using System.IO;

namespace SqliteForWP8
{
    public partial class MainPage : PhoneApplicationPage
    {
        /// <summary>
        /// The database path.
        /// </summary>
        public static string DB_PATH = Path.Combine(Path.Combine(ApplicationData.Current.LocalFolder.Path, "sample.sqlite"));

        /// <summary>
        /// The sqlite connection.
        /// </summary>
        private SQLiteConnection dbConn;

        // Constructor
        public MainPage()
        {
            InitializeComponent();
            /// Define the database path. The sqlite database is stored in a file.
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            /// Create the database connection.
            dbConn = new SQLiteConnection(DB_PATH);
            /// Create the table Task, if it doesn't exist.
            dbConn.CreateTable<Task>();
            /// Retrieve the task list from the database.
            List<Task> retrievedTasks = dbConn.Table<Task>().ToList<Task>();
            /// Clear the list box that will show all the tasks.
            TaskListBox.Items.Clear();
            foreach (var t in retrievedTasks)
            {
                TaskListBox.Items.Add(t);
            }
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            if (dbConn != null)
            {
                /// Close the database connection.
                dbConn.Close();
            }
        }

        private void Insert_Click_1(object sender, RoutedEventArgs e)
        {
            // Create a new task.
            Task task = new Task()
            {
                Title = TitleField.Text,
                Text = TextField.Text,
                CreationDate = DateTime.Now
            };
            /// Insert the new task in the Task table.
            dbConn.Insert(task);
            /// Retrieve the task list from the database.
            List<Task> retrievedTasks = dbConn.Table<Task>().ToList<Task>();
            /// Clear the list box that will show all the tasks.
            TaskListBox.Items.Clear();
            foreach (var t in retrievedTasks)
            {
                TaskListBox.Items.Add(t);
            }
        }
```


(C) Copyright Nokia 2013. All rights reserved.

```
    }

    /// <summary>
    /// Task class representing the Task table. Each attribute in the class become one attribute in the database.
    /// </summary>
    public sealed class Task
    {
        /// <summary>
        /// You can create an integer primary key and let the SQLite control it.
        /// </summary>
        [PrimaryKey, AutoIncrement]
        public int Id { get; set; }

        public string Title { get; set; }

        public string Text { get; set; }

        public DateTime CreationDate { get; set; }

        public override string ToString()
        {
            return Title + ":" + Text + " < " + CreationDate.ToShortDateString() + " " + CreationDate.ToShortTimeString();
        }
    }
}
```

Preparing the application to publish to the Store

If you don't want to publish the application to the Store, the application is finished and you can jump over this section.

For some reason, the current code is not able to create the database file when the application is signed and downloaded from the Store. This is a blocker issue if you want to publish the application to the Store using SQLite databases.

This section describes a step-by-step guide to create the database file programmatically on first application startup and avoid this issue. Basically, what we are going to do is copy the database file from the installation folder (That is read-only and can be updated in development time) to the local folder (That is read-write and can be updated only in runtime) only on first startup. The given example already follow these steps.

Getting database file using the Isolated Storage Explorer tool

First of all, with the current application running on device or emulator, you have to use the Isolated Storage Explorer tool to get the database file.

Normally, this tool is installed in this folder **Program Files (x86)\Microsoft SDKs\Windows Phone\v8.0\Tools\IsolatedStorageExplorerTool**. Go to this folder and run one of the below commands:

- **ISETool.exe ts xd 8a40681d-98fc-4069-bc13-91837a6343ca c:\data\myfiles** command, if you are running the application on emulator.
- **ISETool.exe ts de 8a40681d-98fc-4069-bc13-91837a6343ca c:\data\myfiles** command, if you are running the application on device.

The third argument is the application product ID, you can get it in the **WMAppManifest.xml** file, **App** tag, **ProductId** attribute. The product ID of the given example is **8a40681d-98fc-4069-bc13-91837a6343ca**.

You can see more details about the Isolated Storage Explorer tool here.

Add the database file as a content

Now, if everything is okay, you should have a copy of the Isolated Storage content in the **c:\data\myfiles**; And the database file should be there too. I'm considering that you are running the example shown in this article, so the file name is **sample.sqlite**. In order to copy this file to your project, follow these steps: Rigth click on the *SqliteForWP8* project; Click on *Add* option; Click on *Add Existing Item* option; Select the **c:\data\myfiles\sample.sqlite** file and click *Add*.

Tip: Remember to change the *Copy to Ouput Directory* property of the added **sample.sqlite** file to *Copy if newer*.

Copy database file to local folder on first startup

Remove the `private void Application_Launching(object sender, LaunchingEventArgs e)` method from the `Application` class and use the below one. This method will try to find a database file in the isolated storage, if the file is not found, it'll copy the file from the installation folder to the local folder.

```
// Code to execute when the application is launching (eg, from Start)
// This code will not execute when the application is reactivated
private async void Application_Launching(object sender, LaunchingEventArgs e)
{
 StorageFile dbFile = null;
 try
 {
  // Try to get the
  dbFile = await StorageFile.GetFileFromPathAsync(MainPage.DB_PATH);
 }
 catch (FileNotFoundException)
 {
  if (dbFile == null)
  {
   // Copy file from installation folder to local folder.
   // Obtain the virtual store for the application.
   IsolatedStorageFile iso = IsolatedStorageFile.GetUserStoreForApplication();

   // Create a stream for the file in the installation folder.
   using (Stream input = Application.GetResourceStream(new Uri("sample.sqlite", UriKind.Relative)).Stream)
   {
    // Create a stream for the new file in the local folder.
```


(C) Copyright Nokia 2013. All rights reserved.

```
      using (IsolatedStorageFileStream output = iso.CreateFile(MainPage.DB_PATH))
      {
       // Initialize the buffer.
       byte[] readBuffer = new byte[4096];
       int bytesRead = -1;

       // Copy the file from the installation folder to the local folder.
       while ((bytesRead = input.Read(readBuffer, 0, readBuffer.Length)) > 0)
       {
        output.Write(readBuffer, 0, bytesRead);
       }
      }
     }
    }
   }
  }
```

References

- Working with SQLite in Windows Phone 8: a sqlite-net version for mobile
- sqlite-net-wp8

(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.