Universal System.Data.Sqlite binary for .NET and Mono

Alexei Shcherbakov, 21 Sep 2014

★★★★★ 4.74 (12 votes)

How to make almost universal NuGet 'System.Data.Sqlite' package for .NET and Mono

Download SystemDataSqliteMonoSolution.zip - 45.3 KB

Download NugetPackages.zip - 2.5 MB

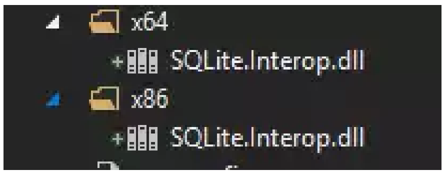
Universal System.Data.Sqlite binary for .NET and Mono

Introduction

System.Data.Sqlite is a wonderfull ADO .NET provider library, but same binary cannot be used on Windows(.NET) and Linux (Mono) simultaneously.

Under Mono/Linux Mono.Data.SQLite provider is exist, but using 2 different providers in same app is a bad idea.

Standard System.Data.Sqlite.Core package (with 2 Interop libs) work perfectly in Windows, but not working in Linux out-of-box. Also it have bad 'feature' - package installer tries to add Interop libs into source control system.



Mixed Mode Library don't have this bad 'feature', but in can run only on one platform and don't run under Linux.

So I try to make simple solution that will work under Windows/.NET (x86/x86_64) and Linux/Mono(x86/x86_64/ARM)

It will use nuget system for deploying resulting packages

How to use packages

Attached Packages

1. Download attached packages
2. Create local NuGet package source and place download packages here
3. Install NuGet packages into your project in Visual Studio or MonoDevelop (since Version 5)

System.Data.Sqlite.Core.MSILstd - Package that contains managed System.Data.Sqlite.dll - must be referenced in any project in solution, that use SQLite library

Use should choice **ONLY** 1 variant:

1. System.Data.Sqlite.SqliteBinaries - Package that contains sqlite3.dll for (x86/x86_64) - must be referenced ONLY in executable project (Application or Test project)
2. System.Data.Sqlite.SqliteBinaries.x86 - Package that contains sqlite3.dll for x86 - must be referenced ONLY in executable project (Application or Test project)
3. System.Data.Sqlite.SqliteBinaries.x86_64 - Package that contains sqlite3.dll for x86_64 - must be referenced ONLY in executable project (Application or Test project)

Logic demonstration with 'sandbox' solution

Sandbox solution

Solution contains 3 project

1. Application - console executable that use System.Data.Sqlite
2. Library - C# library that use System.Data.Sqlite
3. Test - nunit test application that use System.Data.Sqlite

'Application' uses packages: System.Data.Sqlite.Core.MSILstd & System.Data.Sqlite.SqliteBinaries

'Library' uses packages: System.Data.Sqlite.Core.MSILstd

'Test' uses packages: System.Data.Sqlite.Core.MSILstd & System.Data.Sqlite.SqliteBinaries

Application can be runned under Windows, Linux(x86,x86_x64,ARM)

History

Article

1. Building System.Data.Sqlite for Mono

System.Data.Sqlite consists of two parts - managed and unmanaged(interop) parts. They can be combined in single mixed assembly in Microsoft .NET. We need to change System.Data.Sqlite to use standard sqlite3 library

First - download System.Data.Sqlite sources to make package. Current version is **1.0.93.0(3.8.5)**

After download it must be compiled. Standard compile procedure is:

1. Navigate to Setup directory
2. Choice .NET version
    - .NET 2.0 - **set_2005.bat**
    - .NET 3.5 - **set_2008.bat**
    - .NET 4.0 - **set_2010.bat**
    - .NET 4.5 - **set_2012.bat**
    - .NET 4.5.1 - **set_2013.bat**
3. Setup build parameters ("**SET MSBUILD_ARGS=/property:UseInteropDll=false /property:UseSqliteStandard=true**")
4. Build it - "**build.bat ReleaseManagedOnly**"

I placed resulting library into System.Data.Sqlite.Core.MSILstd package and attached it to this article

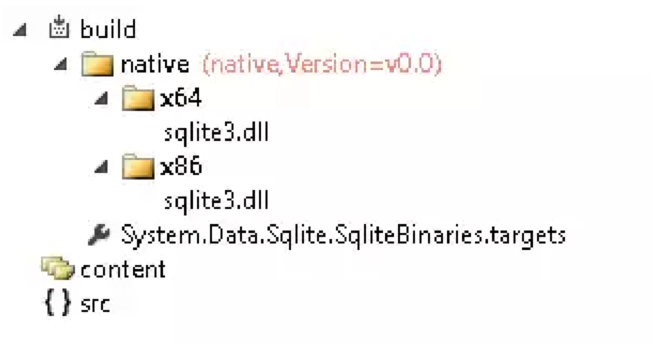
But this is only managed part of library. Also we need to make deploy procedure for sqlite3.dll because library can't work without it unmanaged part (in linux sqlite can be installed as package, but under windows it must 'cames' with application)

Simpliest way to create package - use NuGet Package Explorer, open System.Data.Sqlite.Core.MSIL package and replace dlls with created ones.

2. NuGet deploy procedure for Sqlite library

NuGet has special feature for coping files to project output directory at build. We can use it to copy sqlite3.dll (x86 & x86_64) to output dir

'System.Data.Sqlite.SqliteBinaries' Package structure:



```
build
    native (native,Version=v0.0)
        x64
            sqlite3.dll
        x86
            sqlite3.dll
    System.Data.Sqlite.SqliteBinaries.targets
content
{ } src
```

Dll's are placed into build/native NuGet folder in package. After, special .target file(with name of package) with deploy logic was created

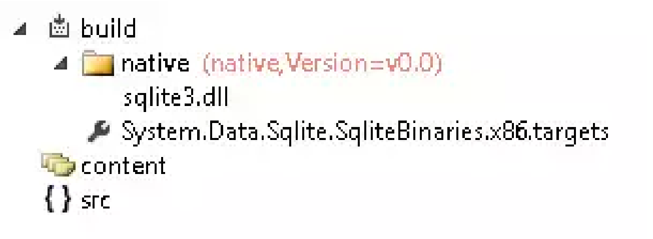
System.Data.Sqlite.SqliteBinaries.targets contents:

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
    <PropertyGroup>
        <BuildDependsOn>
            copy_sqlite3_ALL_to_outputpath;
            $(BuildDependsOn);
        </BuildDependsOn>
    </PropertyGroup>
    <Target Name="copy_sqlite3_ALL_to_outputpath">
        <Message text = "Copying sqlite3.dll (x86 and x86_x64)" />
        <Copy SourceFiles="$(MSBuildThisFileDirectory)/native/x86/sqlite3.dll"
DestinationFolder="$(OutputPath)/x86" />
        <Copy SourceFiles="$(MSBuildThisFileDirectory)/native/x64/sqlite3.dll"
DestinationFolder="$(OutputPath)/x64" />
    </Target>
</Project>
```

After successeful build NuGet will automatically copy sqlite3.dll binaries to output directory

Also package can be created for specific architecture. It is more simple.

System.Data.Sqlite.SqliteBinaries.x86 package structure:



```
build
    native (native,Version=v0.0)
        sqlite3.dll
    System.Data.Sqlite.SqliteBinaries.x86.targets
content
{ } src
```

System.Data.Sqlite.SqliteBinaries.x86.targets contents:

```
<span style="color: #8b008b"><?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
    <PropertyGroup>
        <BuildDependsOn>
            copy_sqlite3_x86_to_outputpath;
            $(BuildDependsOn);
        </BuildDependsOn>
    </PropertyGroup>
    <Target Name="copy_sqlite3_x86_to_outputpath">
        <Message text = "Copying sqlite3.dll x86" />
        <Copy SourceFiles="$(MSBuildThisFileDirectory)/native/sqlite3.dll"
DestinationFolder="$(OutputPath)"/>
    </Target>
</Project></span>
```

System.Data.Sqlite.SqliteBinaries.x86_64 package have same structure except .target name - System.Data.Sqlite.SqliteBinaries.x86_64.targets

3. Testing resulting solution

Dev & test platforms

1. PC - Windows 8.1/.NET 4.5.2 (x86_64) with Visual Studio 2013 Update 3 and Xamarin Studio 5(same as MonoDevelop 5)
2. PC - OpenSuse Factory/Mono (x86) with MonoDevelop 5
3. Home Server - CentOS 6/Mono (x86_64 and x86 on KVM)
4. BeagleBone Black - Debian/Mono (ARM)

Unfortunately I haven't any Linux/Mono (x86_64) machine for tests. I hope that if solution works for Mono on x86 & ARM architecures - it will work for x86_64 😊

Test database have 2 tables (Pets & Houses) and 3 entities (Cat,Dog & House). Logic is quiet simple. Cat is Pet, Dog is Pet, Pets live in House, House have address. This sample use NHibernate as ORM library.

Application project creates database, add some test data, then query results

Tests project demonstrate atomic operations.

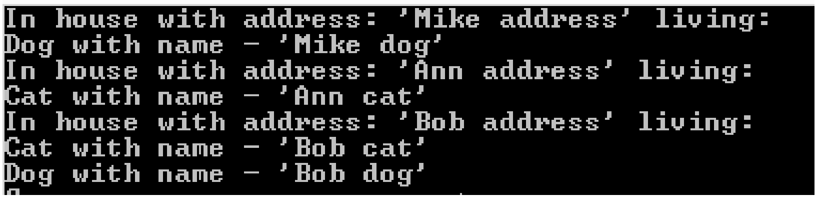
Application & Test projects use Library project

4. Test results under different platforms

4.1. PC/Windows 8.1 x86_64/.NET/VisualStudio 2013 Update 3

To integrate nunit into VS2013 a special package must be installed ('NUnit Test Adapter') - you can see nunit tests inside 'Test Explorer' after it installation. Sample code includes 'console runner' so it can be executed even 'NUnit Test Adapter' is not installed

Application working normally

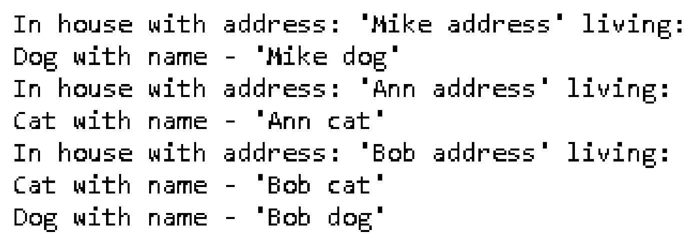


```
In house with address: 'Mike address' living:
Dog with name - 'Mike dog'
In house with address: 'Ann address' living:
Cat with name - 'Ann cat'
In house with address: 'Bob address' living:
Cat with name - 'Bob cat'
Dog with name - 'Bob dog'
```

4.2. PC/Windows 8.1 x86_64/.NET 4.5.2/Xamarin Studio 5

Warning: on .NET x86_64 debugging is not working in MonoDevelop/Xamarin Studio

Application working normally in 'Run' mode (Debug mode on .NET x86_64 is not working in MonoDevelop)

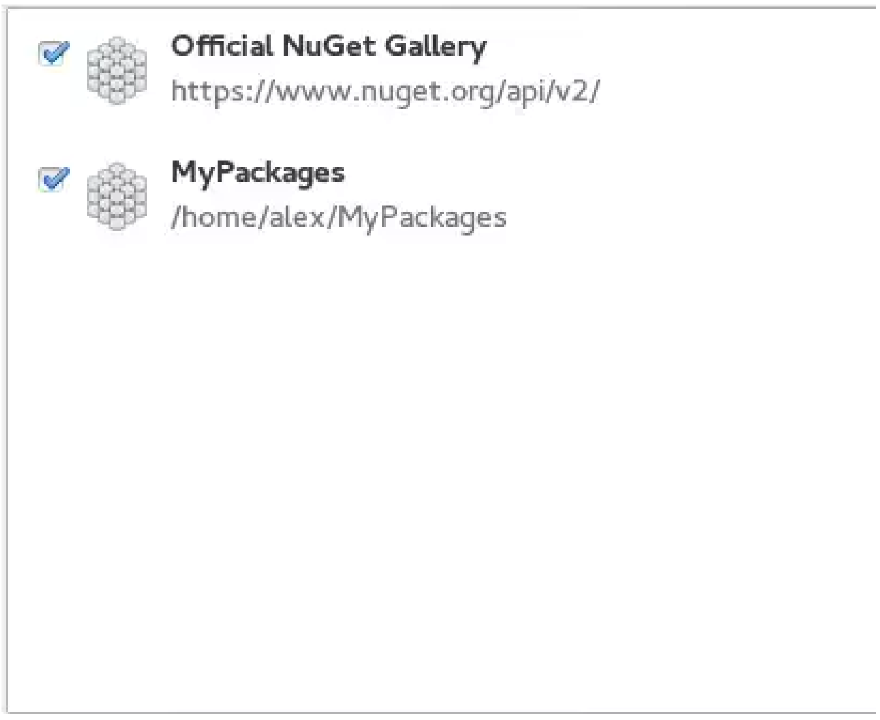


```
In house with address: 'Mike address' living:
Dog with name - 'Mike dog'
In house with address: 'Ann address' living:
Cat with name - 'Ann cat'
In house with address: 'Bob address' living:
Cat with name - 'Bob cat'
Dog with name - 'Bob dog'
```

4.3. PC/OpenSUSE Factory x86/Mono/MonoDevelop 5

Sqlite3 Native library version - 3.8.5

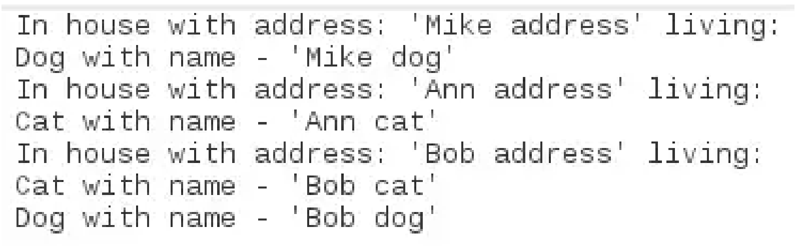
At first we need add our packages to local package source of MonoDevelop



Official NuGet Gallery
https://www.nuget.org/api/v2/

MyPackages
/home/alex/MyPackages

After it - restore and build. Application is also working normally



```
In house with address: 'Mike address' living:
Dog with name - 'Mike dog'
In house with address: 'Ann address' living:
Cat with name - 'Ann cat'
In house with address: 'Bob address' living:
Cat with name - 'Bob cat'
Dog with name - 'Bob dog'
```

4.4.a. Home File Server/CentOS 6 x86_64/Mono/Run in console

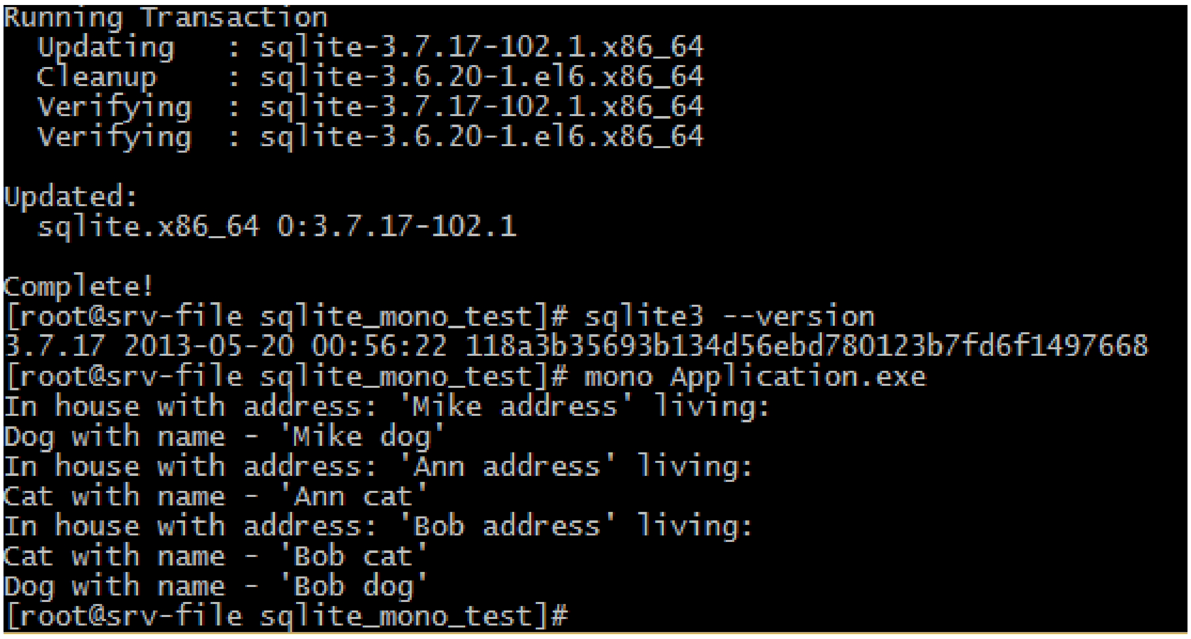
Mono version - 3.10.1 (master/ca51c3b) - Fresh
Sqlite3 Native library version - 3.6.20

Not working. Exception 'SQL logic error or missing database'

May be SQLite version is too old?

Googling on 'CentOS SQLite 3.7' gives us link:
ftp://ftp.pbone.net/mirror/ftp5.gwdg.de/pub/opensuse/repositories/Application:/Geo/CentOS_6/x86_64/sqlite-3.7.17-102.1.x86_64.rpm

Now we have:



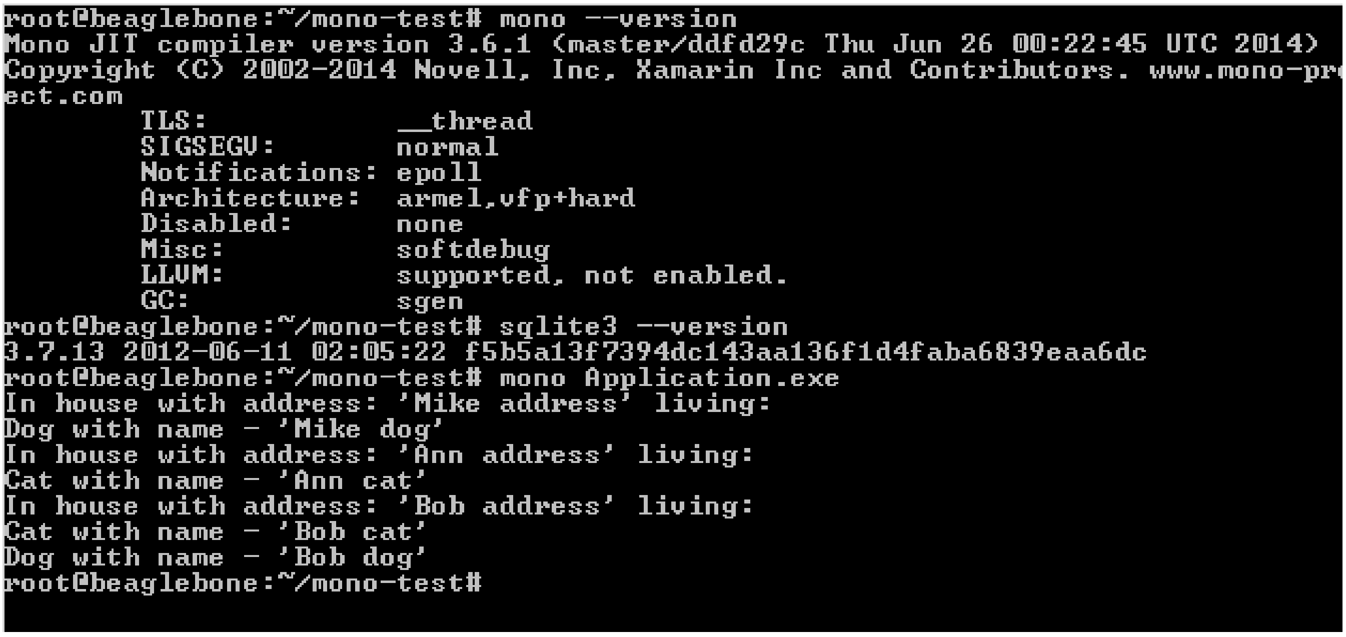
```
Running Transaction
  Updating   : sqlite-3.7.17-102.1.x86_64
  Cleanup    : sqlite-3.6.20-1.el6.x86_64
  Verifying  : sqlite-3.7.17-102.1.x86_64
  Verifying  : sqlite-3.6.20-1.el6.x86_64

Updated:
  sqlite.x86_64 0:3.7.17-102.1

Complete!
[root@srv-file sqlite_mono_test]# sqlite3 --version
3.7.17 2013-05-20 00:56:22 118a3b35693b134d56ebd780123b7fd6f1497668
[root@srv-file sqlite_mono_test]# mono Application.exe
In house with address: 'Mike address' living:
Dog with name - 'Mike dog'
In house with address: 'Ann address' living:
Cat with name - 'Ann cat'
In house with address: 'Bob address' living:
Cat with name - 'Bob cat'
Dog with name - 'Bob dog'
[root@srv-file sqlite_mono_test]#
```

It is working. Take into account that 'System.Data.SQLite' will not work with very old version of sqlite3 library

4.5. BeagleBone Black/Debian ARM/Mono/Run in console

Mono version - 3.6.1 (master/ddfd29c)
Sqlite3 Native library version - 3.7.13

Application working normally



```
root@beaglebone:~/mono-test# mono --version
Mono JIT compiler version 3.6.1 (master/ddfd29c Thu Jun 26 00:22:45 UTC 2014)
Copyright (C) 2002-2014 Novell, Inc, Xamarin Inc and Contributors. www.mono-pr
ect.com
        TLS:           __thread
        SIGSEGV:       normal
        Notifications: epoll
        Architecture:  armel,vfp+hard
        Disabled:      none
        Misc:          softdebug
        LLVM:          supported, not enabled.
        GC:            sgen
root@beaglebone:~/mono-test# sqlite3 --version
3.7.13 2012-06-11 02:05:22 f5b5a13f7394dc143aa136f1d4faba6839eaa6dc
root@beaglebone:~/mono-test# mono Application.exe
In house with address: 'Mike address' living:
Dog with name - 'Mike dog'
In house with address: 'Ann address' living:
Cat with name - 'Ann cat'
In house with address: 'Bob address' living:
Cat with name - 'Bob cat'
Dog with name - 'Bob dog'
root@beaglebone:~/mono-test#
```

Conclusion

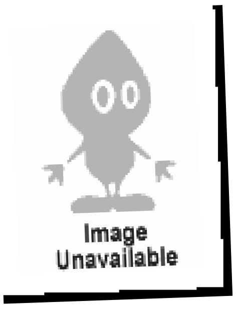
Remember! This solution MAY not normally run for NuGet prior 2.7 (Package restore issues, after 2.7 version this feature is enabled by default). Also it MAY not work MonoDevelop prior version 5(It have bad NuGet integeration). Also System.Data.Sqlite will not work with very old linux versions of SQLite (1.0.93.0(3.8.5) is working with 3.7.13, but not working with 3.6.20) so if you plan support system with old SQLite versions (like CentOS) you must include latest precompiled native SQLite library with your app.

Suggested solution works on all modern developer platforms. Also you can adopt it for other cases when you need deploy native libraries inside NuGet package

License

Share

About the Author



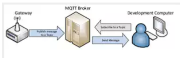
Alexei Shcherbakov No Biography provided
Software Developer
Russian Federation

You may also be interested in...

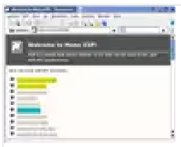
Introduction to Mono - Your first Mono app

Optimize SharePoint Storage with BLOB Externalization

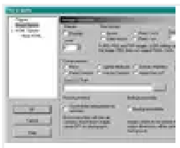
Case-Insensitive Sort of UTF8 Data Using System.Data.SQLite



Intel® IoT Gateways: Publishing Data to an MQTT Broker Using Python



Introduction to Mono - ASP.NET with XSP and Apache



SAPrefs - Netscape-like Preferences Dialog

Comments and Discussions

9 messages have been posted for this article Visit **http://www.codeproject.com/Articles/821149/Universal-System-Data-Sqlite-binary-for-NET-and-Mo** to post and view comments on this article, or click **here** to get a print view with messages.