Including SQLite.Interop.dll into your C# project

When developing C# applications I love to be able to distribute a single executable without having to install the software or worry about dependencies.

Whenever I create a new C# project which is going to refer to external libraries such as the wast popular Html Agility Pack, I normally start by adding Costura to my project using NuGet.

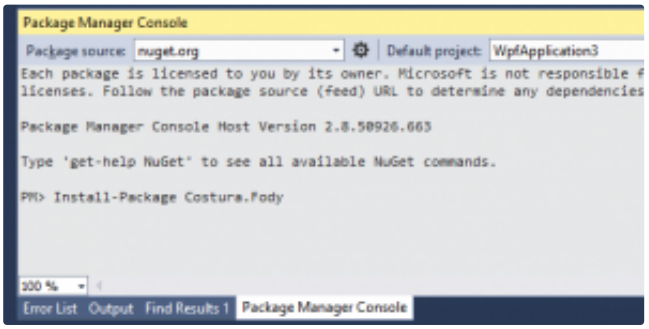
If you haven't used NuGet before I highly recommend you to do so! It's just awesome and makes handling decencies much easier in terms of keeping track of them and dependencies, updating and building. Normally it's enough to just add Costura to your project and all of the external libraries will automatically be embedded into your final executable upon build.

However, when I started to work with SQLite.Interop.dll I wasn't able to include the library. This guide will show you what I had to do in order to solve this issue. This will most likely work for all other libraries that are targeted to a specific platform (32- and 64-bits).

Create a new project

Start by creating a new C# project. For this tutorial I'm creating a new **WPF Application** using **.Net Framework 4.5**.

If you don't have the**Package Manager Console**open, go to View->Other Windows->Package Manager Console in order to open it.



In order to install Costura and SQLite run

the following two commands (one after the other):

```
Install-Package Costura.Fody
Install-Package System.Data.SQLite

```

Once you build the project you'll notice that SQLite creates a x86 and x64 directory in /bin/Debug and a whole bunch of other DLL files. You should also notice that your **WpfApplication.exe** is about 1894 KB in size. Just as a reference before and after embedding SQLite.

Embedding SQLite.Interop.dll

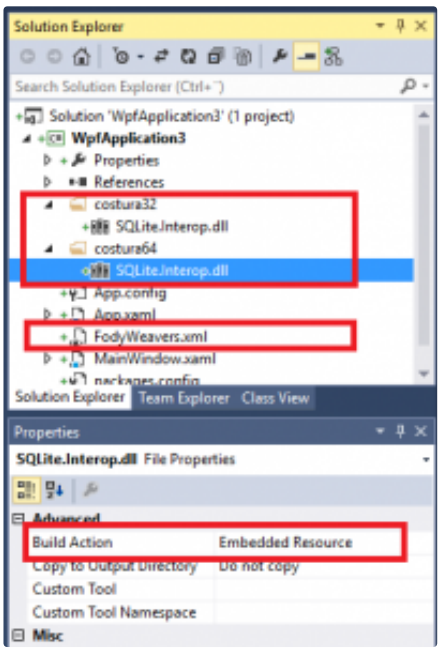
Now comes the fun part. Let's get rid of all the external dependencies! After installing Costura a files called **FodyWeavers.xml** was added to your project. Normally you don't have to specify what DLL files to embed, but this time we do.

Make sure your xml file looks like this:

```
<?xml version="1.0" encoding="utf-8"?>
<Weavers>
 <Costura>
 <Unmanaged32Assemblies>
 SQLite.Interop
 </Unmanaged32Assemblies>
 <Unmanaged64Assemblies>
 SQLite.Interop
 </Unmanaged64Assemblies>
 </Costura>
</Weavers>
```

Now we have specified that we want to embed the DDL-files, but we also need to include them into our project. Next create to new folders called **costura32**and **costura64** and make sure to copy the correct version of **SQLite.Interop.dll** into each of them. Put simply, copy the dll-file from x86 to costura32 and x64 to costura64. You will also have to change the**Build Action** for both files to **Embedded Resource**.

Once done rebuild your project and notice that your executable now is quite a bit larger then it was before. Now it’s safe to delete or move your executable outside the debug directory without breaking any dependencies.



Optional: Clean output folder automatically

Costura only merges dependencies. It does not handle cleaning those dependencies from your output directory. If you want to clean this directory automatically after each build you can install the following cmdlet:

```
Install-CleanReferencesTarget
```