Lab 02: PCL references

Prerequisites

You will need a development environment, either a Mac or Windows PC with the iOS or Android SDK (or both, depending on which platform you prefer to use) and Xamarin tools installed.

Follow the instructions in **Using Nuget with Xamarin Studio.pdf** to install NuGet in Xamarin Studio.

Downloads

https://university2.xamarin.com/materials/xam300-advanced-cross-platform-development

Lab Goals

The goal of this lab will be to:

- Install a PCL-based NuGet package into our own PCL.
- Reference the PCL in platform-specific application projects.
- Add platform-extension libraries that enable the PCL to perform platform-specific functionality.

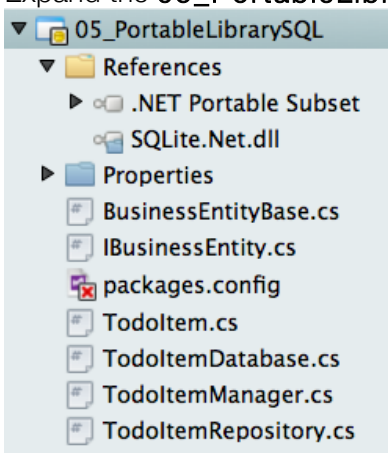
The library we are going to work with is the **SQLite.NET PCL** library available via NuGet.

Steps

Open the Starting Solution

1. Launch your IDE (either Xamarin Studio or Visual Studio)
2. Open the **PortableLab2.sln** solution

3. Expand the **05_PortableLibrarySQL** project

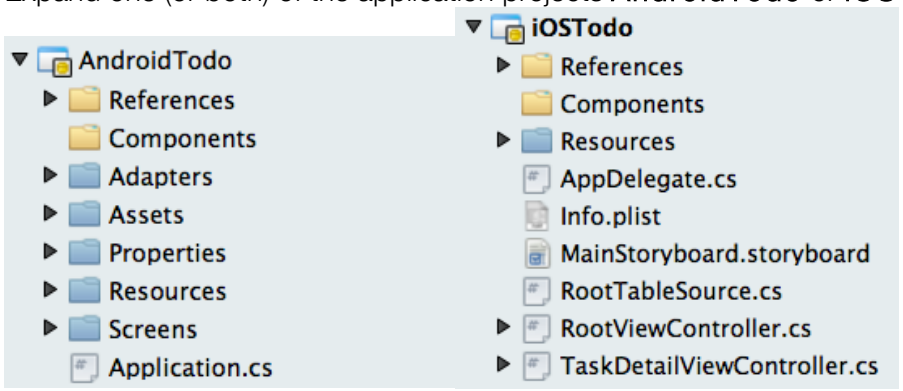


There should be a **packages.config** item which specifies which NuGet packages are required, and the **SQLite.Net.dll** should already be referenced.

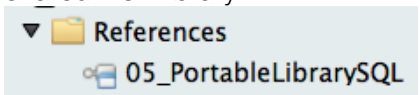
4. Open **TodoItemManager.cs** – the constructor expects an `SQLiteConnection` object to be supplied. This class must be created for a specific platform as it must have read/write access to the filesystem (among other things).

```
public TodoItemManager (SQLiteConnection conn)
{
    repository = new TodoItemRepository(conn);
}
```

5. Expand one (or both) of the application projects **AndroidTodo** or **iOSTodo**

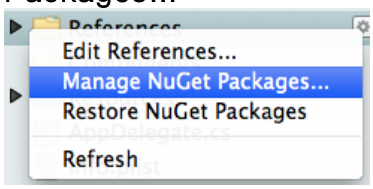


6. Expand the **References** node and notice that they both already reference the shared PCL library

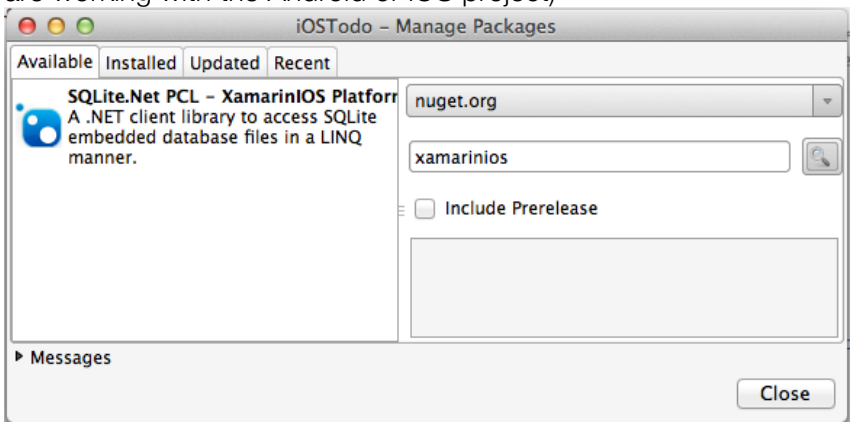


Neither application references any NuGet package though, so we cannot create an instance of **SQLiteConnection** that is required by our PCL.

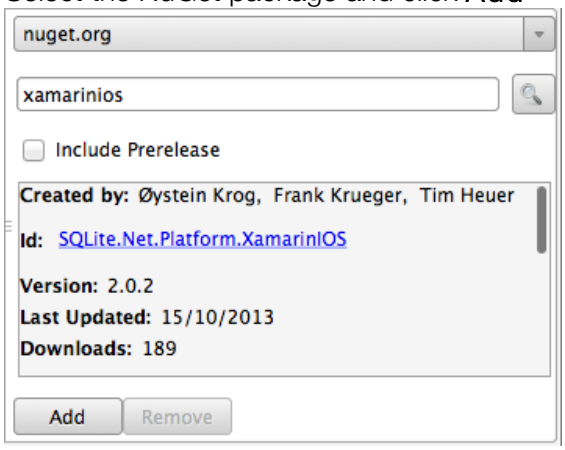
7. Right-click on the application project and choose **Manage NuGet Packages...**



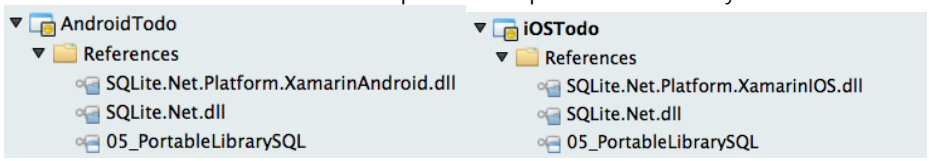
8. Search for either "xamarinandroid" or "xamarinios" (depending on whether you are working with the Android or iOS project)



9. Select the NuGet package and click **Add**



This will add a **packages.config** file to the project and also automatically add references to a PCL and a platform-specific assembly:



10. Now we can wire up the PCL – choose either the Android or iOS code below. You can navigate to the relevant section of the Android **Application.cs** or iOS **AppDelegate.cs** file from the Tasks pad:

```
// TODO: Android Step5: SQLite (don't forget to add NuGet).
```

```
var plat = new SQLite.Net.Platform.XamarinIOS.SQLitePlatformAndroid();
var conn = new SQLite.Net.SQLiteConnection(plat, path);
TaskMgr = new TodoItemManager(conn);
```

OR

// TODO: iOS Step5 SQLite (don't forget to add NuGet).

```
var plat = new SQLite.Net.Platform.XamarinIOS.SQLitePlatformIOS();
var conn = new SQLite.Net.SQLiteConnection(plat, path);
TaskMgr = new TodoItemManager(conn);
```

These lines of code instantiate a platform-specific class `plat` that is defined in the platform-specific library we referenced.

The component author needed to write this class in a platform-specific way so that they could include file-system operations and other code that cannot be written in a PCL.

We then pass the `plat` class to the `SQLiteConnection` which is a class defined in the PCL. We have used the `SQLiteConnection` class (and its associated PCL classes) throughout our own PCL library.

We have "injected" the platform-specific code we need into our PCL to maximize code-sharing while still providing the required functionality on each platform.

Summary

In this lab we implemented a Portable Class Library (PCL) from Nuget in our own custom PCL. We then injected a platform-specific class via a constructor on one of our PCL classes.

You can examine the source of the SQLite.NET PCL library here:

https://github.com/oysteinkrog/SQLite.Net-PCL

and read more about the NuGet package here:

http://www.nuget.org/packages/SQLite.Net-PCL/