Intro to Cross-Platform Mobile Development Xamarin University

Exercise 1: Working with Shared Projects

Duration:

20 minutes

Lab Goals

The primary goal of this exercise will be to load some JSON data in from a file and use it as a source to display information in an iOS, Android and Windows Phone application. The code to manage the loading of the file will be completely shared between the projects using a Shared Project.

We will need to do the following steps

1. Create a new Shared Project.
2. Add 2 shared source files (**Song.cs** and **SongLoader.cs**) to the shared project - these have already been written for you and are in the lab resources folder.
3. Add a reference to the Shared Project to each of your target projects (see the above note about Visual Studio)
4. Add a Nuget reference to Newtonsoft's Json.net parser which is used by the shared project code into each target project.
5. Add a data file into each of the target projects which is processed by the shared code.

Hide

Android Assets
You can store data files, fonts and other non-graphical pieces of information in the **Assets** folder.
Learn more about Using Android Assets.

iOS Resources
iOS supports adding any type of resource file into your project through the **Resources** folder.
Learn more about Working with iOS Resources.

Windows Phone Resources
Windows Phone (WinPRT) supports reading and writing data from the local file system through a new set of classes.
Read more about Working with Files and Folders in Windows Phone 8.1.

6. Modify the shared code to properly load the file based on the platform.

Required Assets

There are included resources in the **Part 01 Resources** folder which are needed to complete this exercise. These are included with the course materials, so make sure you have them available before starting.

Exercise Challenge

Using the above high level steps, try to accomplish the lab goals on your own. If you need some additional help, there are step-by-step instructions below which you can use if you get stuck or need a hint.

Steps

Below are the step-by-step instructions to implement the exercise.

Select the IDE environment you want to work with: XAMARIN STUDIO or VISUAL STUDIO

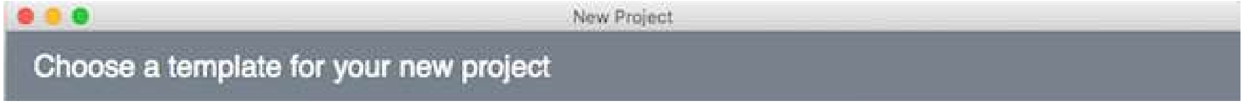
Note for Windows users: If you are using Xamarin Studio on Windows, the only project you can use is the Android one, the below instructions assume Xamarin Studio on the Mac, but the functionality (beyond working with iOS) should be identical.

Open the Starter Solution

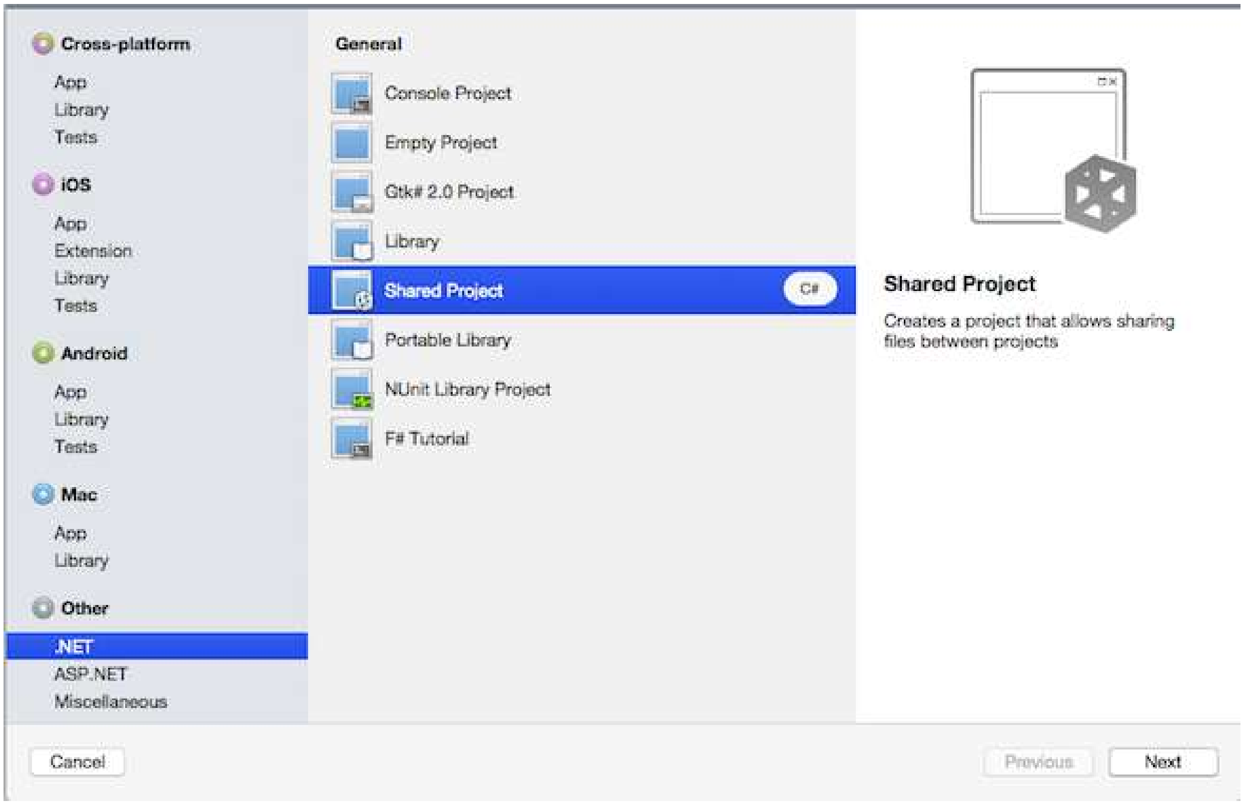
1. Open the starting project **MyTunes** included in the **Part 01 Resources** folder with Xamarin Studio on the Mac.
2. You will be able to run the iOS and Android projects in this development environment. The Windows Phone project will be disabled (grayed out). The two available projects will present a list of data when run.

Add the Shared Project

1. Right click on the Solution node and select **Add > Add a New Project**.
2. Select Shared Project - it can be found in the **Other > .NET** category. Click **Next** to continue.



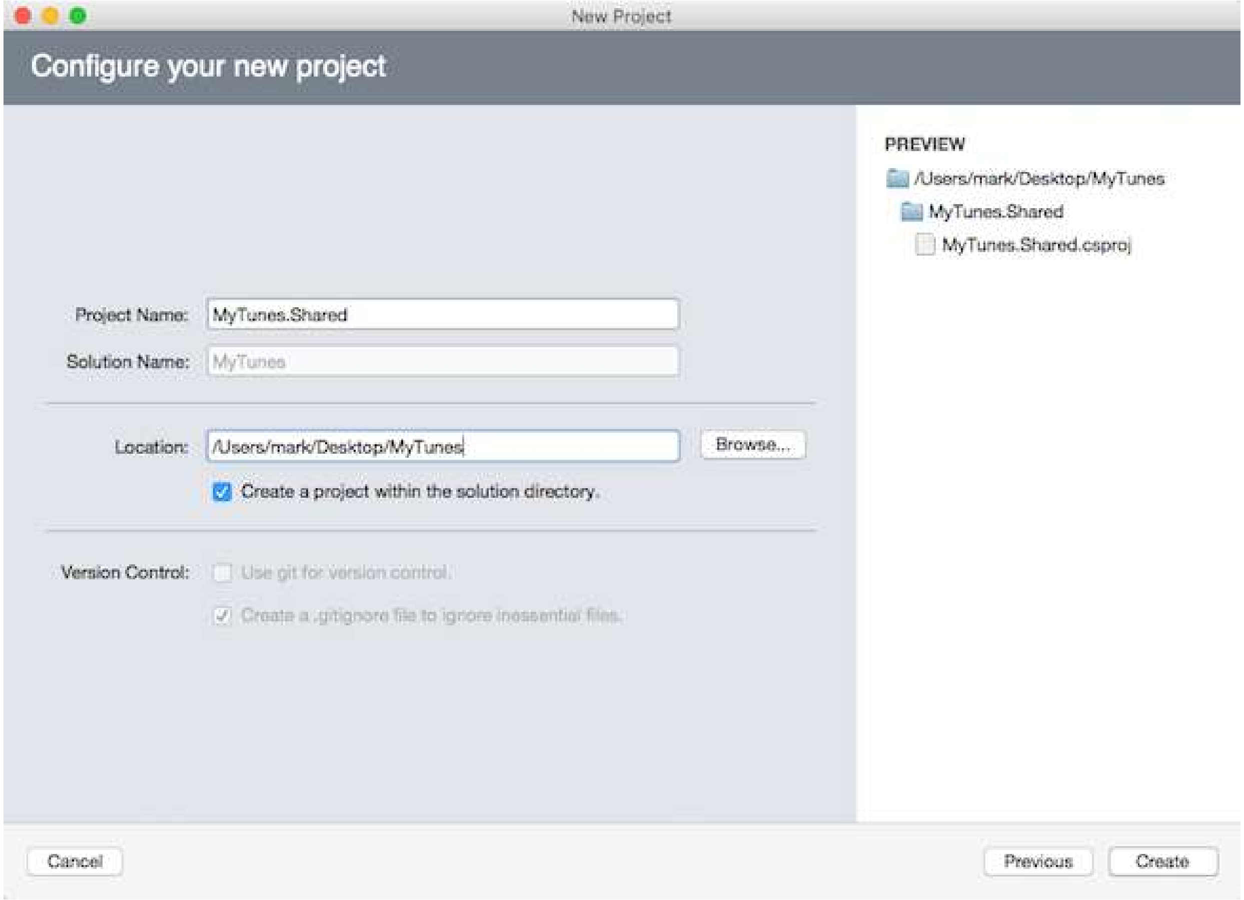
Choose a template for your new project



Shared Project

Creates a project that allows sharing files between projects

3. Name the project "MyTunes.Shared" and click **Create** to create the project.



/Users/mark/Desktop/MyTunes
MyTunes.Shared
MyTunes.Shared.csproj

Project Name: MyTunes.Shared

Solution Name: MyTunes

Location: /Users/mark/Desktop/MyTunes

☑ Create a project within the solution directory.

Version Control: □ Use git for version control.

☑ Create a .gitignore file to ignore inessential files.

4. You can delete the template C# file added to the project if there is one. We will replace it with some pre-written files.

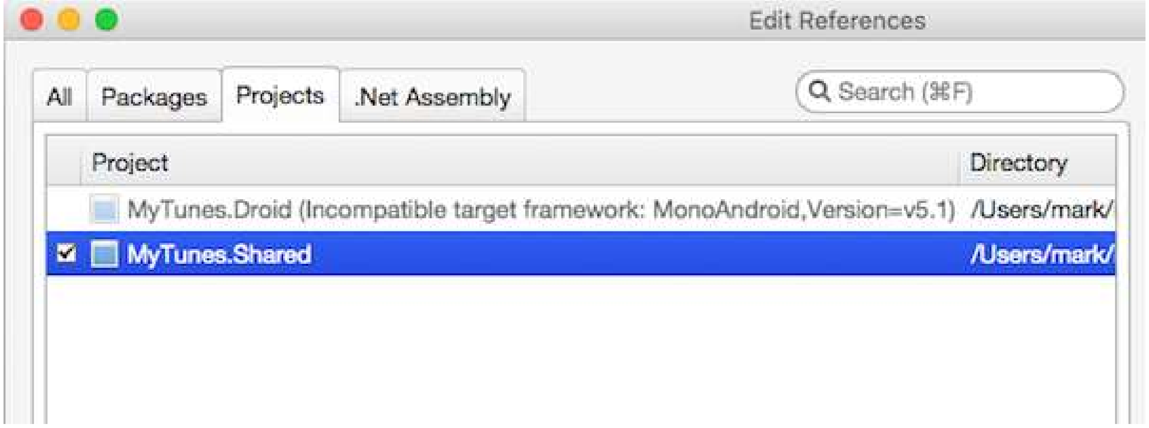
Add the code into the Shared Project

Add the Code into the Shared Project

1. In the **Part 01 Resources** folder there is a **Data** sub-folder with two C# source files:
    - **Song.cs** - this defines a class which provides information about a single song.
    - **SongLoader.cs** - this is a class which can parse a Json file off disk into a set of `Song` objects. It utilizes the Newtonsoft Json.net parser.
2. Add both C# files to your shared project - you can add existing files to shared projects in the normal fashion, right-click on the project node in the Solution explorer and use the **Add > Add Files...** option. Navigate to the **Data** folder and add both files.
3. Look at both source files to get a sense of the content you will be working with.

Add a Reference to the Shared Project

1. You can add a reference to the Shared Project like any other reference - just right-click on the **References Folder** and select **Edit References**.
2. In the references dialog, select the **Projects** tab and then select the Shared Project in the list.

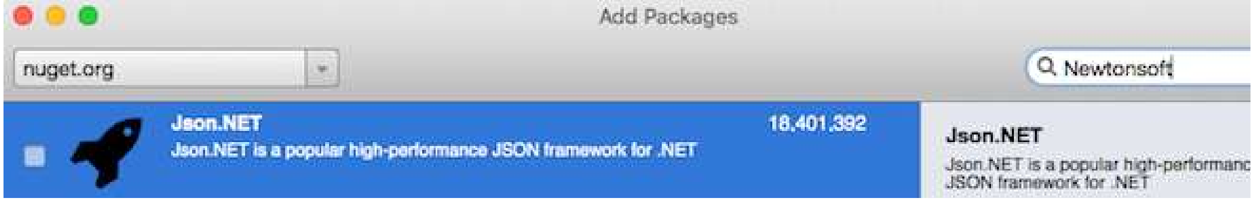


	Project	Directory
	MyTunes.Droid (Incompatible target framework: MonoAndroid,Version=v5.1)	/Users/mark/
☑	MyTunes.Shared	/Users/mark/

3. Do these steps for each of the projects you want to work with (iOS | Android).
4. Try to build the projects - it should give you compile errors because we don't have a reference to the Newtonsoft Json.net parser which is used by the shared code.
5. If you open the **SongLoader.cs** file, you will see that the identifiers are all colored in Red.

Add a Nuget reference to Json.net

1. Open the Nuget dialog by right-clicking on the **Packages** folder in each of your platform projects you want to work with (iOS | Android) and selecting **Add Packages...**.
2. Search online and find the Newtonsoft Json.net component - it should be one of the first packages displayed due to it's popularity.

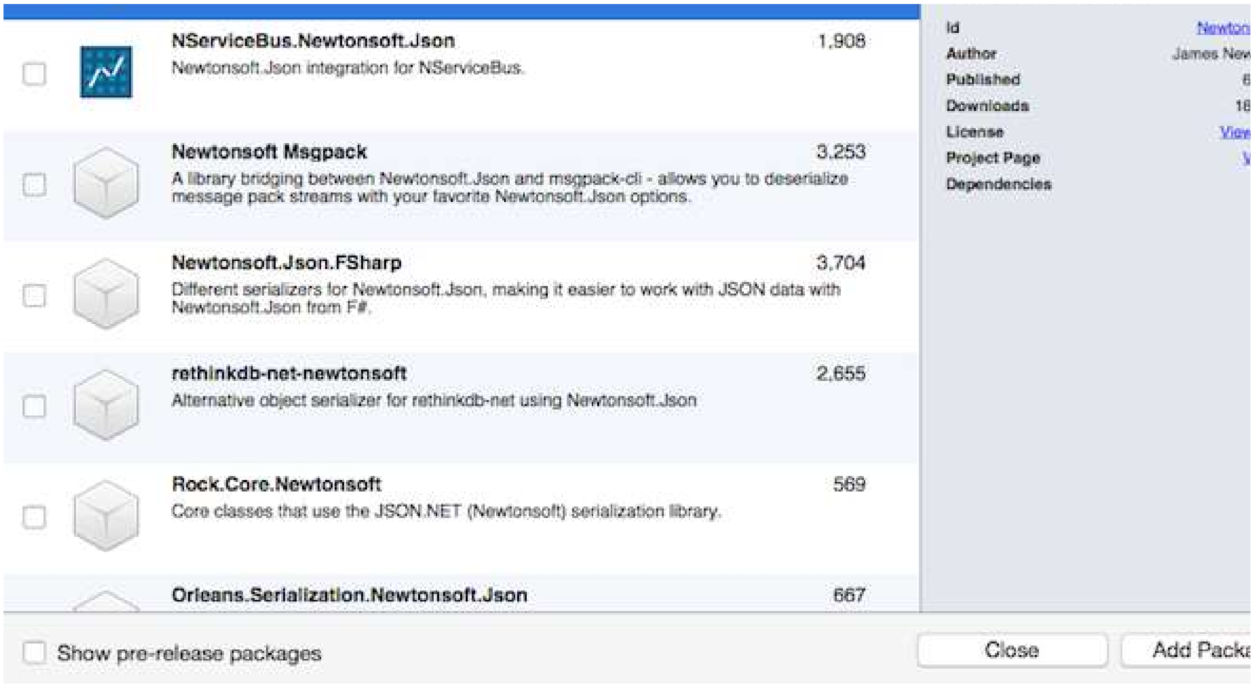


nuget.org

Newtonsoft

Json.NET
Json.NET is a popular high-performance JSON framework for .NET
18,401,392

Json.NET
Json.NET is a popular high-performanc
JSON framework for .NET



□	Package	Downloads
□	**NServiceBus.Newtonsoft.Json** Newtonsoft.Json integration for NServiceBus.	1,908
□	**Newtonsoft Msgpack** A library bridging between Newtonsoft.Json and msgpack-cli - allows you to deserialize message pack streams with your favorite Newtonsoft.Json options.	3,253
□	**Newtonsoft.Json.FSharp** Different serializers for Newtonsoft.Json, making it easier to work with JSON data with Newtonsoft.Json from F#.	3,704
□	**rethinkdb-net-newtonsoft** Alternative object serializer for rethinkdb-net using Newtonsoft.Json	2,655
□	**Rock.Core.Newtonsoft** Core classes that use the JSON.NET (Newtonsoft) serialization library.	569
	Orleans.Serialization.Newtonsoft.Json	667

Id Newton
Author James New
Published 6
Downloads 18
License View
Project Page V
Dependencies

3. Add the package to each of your supported platforms - the package must be added into the *platform-specific* projects, notice that the Shared Project does not even have a references node!
4. Look at the `SongLoader` class again - notice it now is color coded properly because the reference has been added. The project should also compile now.

Use the Song Loader

Now, lets's add some code into each project to use the SongLoader to populate the UI. Pick each platform you want to run and use the steps below to adjust the code.

iOS

1. Locate the `ViewDidLoad` method in the `MyTunesViewController`.
2. Comment out the existing `TableView.Source` assignment.

> The `ViewControllerSource` being used here is a simple, generic, `UITableViewSource` that uses delegates to identify the text and detail text for a row. You can examine the source code in the corresponding source file.

3. Make a call to load the data using `SongLoader.Load`. This method is async, so you will need to decorate the method with `async` and use the `await` keyword.
4. Take the resulting data and turn it into a `List<Song>`.
5. Create a new `ViewControllerSource<Song>` and assign the following properties:
   - `DataSource` property to your new list.
   - `TextProc` to a lambda that returns the name: `s => s.Name`.
   - `DetailTextProc` to a lambda that returns the artist and album: `s => s.Artist + " - " + s.Album`.
6. Assign the table source to the `TableView.Source` property.

Show Code

Android

Android

1. Locate the `OnCreate` method in the `MainActivity`.
2. Comment out the existing `ListAdapter` assignment.

   > The `ListAdapter` being used here is a simple, generic, `BaseAdapter` that uses delegates to identify the text and detail text for a row. You can examine the source code in the corresponding source file.

3. Make a call to load the data using `SongLoader.Load`. This method is async, so you will need to decorate the method with `async` and use the `await` keyword.
4. Take the resulting data and turn it into a `List<Song>`.
5. Create a new `ListAdapter<Song>` and assign the following properties:
   - `DataSource` property to your new list.
   - `TextProc` to a lambda that returns the name: `s => s.Name`.
   - `DetailTextProc` to a lambda that returns the artist and album: `s => s.Artist + " - " + s.Album`.
6. Assign the object to the `ListAdapter` property.

Show Code

Windows Phone

1. Locate the `OnNavigatedTo` method in the `MainPage`.
2. Make a call to load the data using `SongLoader.Load`. This method is async, so you will need to decorate the method with `async` and use the `await` keyword.
3. Take the resulting data and assign it to the `DataContext` property.
4. There is already a `DataTemplate` setup to display the song data.

Show Code

Make sure all your code compiles. It will fail at runtime because we don't have the data yet - let's do that next.

Add the songs data file

We will add the **songs.json** file from the **Part 01 Resources** folder into each target platform project - storing it into the normal resources area for each.

1. **iOS** - Add the **songs.json** file into the **Resources** folder and make sure the build action is marked as "BundleResource"
2. **Android** - Add the **songs.json** file into the **Assets** folder and make sure the build action is marked as "AndroidAsset"
3. **Windows Phone** - Add the **songs.json** file into the root of the project (so the filename will match) and set the build action to "Content"
4. Make sure all your projects still build succesfully.

Add code to read the file

Since we've got the data file stored in a platform-specific way, we will need to use platform-specific code to get to our data. We can use any of the three approaches outlined in the class session code to get to our data. We can use any of the three approaches outlined in the class session (conditional compilation, cloning and partial classes), and you should experiment and try each one in turn; however for the sake of time, we will use conditional compilation in these instructions.

All our work will be done in the `SongLoader` class, specifically the existing `OpenData` method which should open the file (the `Filename` constant defined in the file) and return a `System.IO.Stream`.

Pick the instructions for your target platforms you want to support. Note that if you work with Windows Phone, it will change some signatures because we are using the WinPRT APIs which have a completely new I/O API.

iOS

1. Add a conditional marker for the iOS code. Xamarin.iOS defines the `__IOS__` symbol for this.
2. In your conditional code block, use the `System.IO.File.OpenRead` method to open the filename.
3. Make sure the iOS project builds.

Show Code

Android

1. Add a conditional marker for the Android code. Xamarin.Android defines the `__ANDROID__` symbol for this.
2. In your conditional code block, use the `Android.App.Application.Context.Assets.Open` method to open the filename.
3. Make sure the Android project builds.

Show Code

Windows Phone

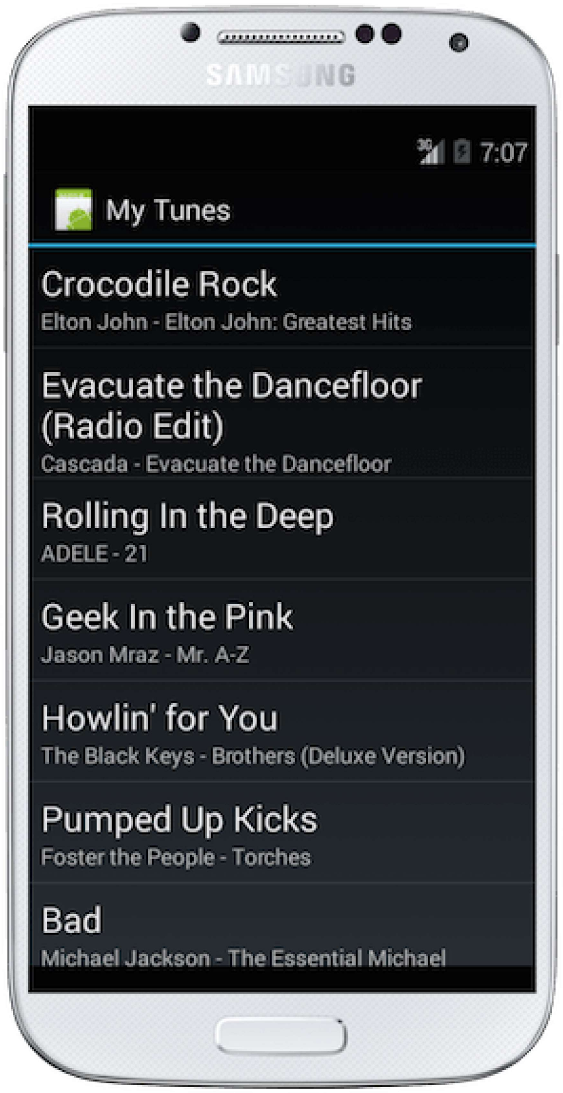
1. Add a conditional marker for the Windows Phone code. Microsoft defines the `WINDOWS_PHONE_APP` symbol for this.
2. In your conditional code block, use the `Windows.ApplicationModel.Package.Current.InstalledLocation.GetFileAsync` method to open the filename. This returns a `StorageFile`.
3. Call `OpenStreamForRead` on the `StorageFile` to retrieve a `Stream`.
4. Because WinPRT APIs are all async, you will need to change the app signature to be `async` and use the `await` keyword - the call site will also need to be adjusted for this.

5. The other platform code can remain the way it is - you will get a warning from the compiler because no `await` keyword is used even though the method is decorated with `async`.
6. Make sure the Windows Phone project builds.

Show Code

Test the applications

1. Run each of the platforms - they should all display the song list.



My Tunes

Crocodile Rock
Elton John - Elton John: Greatest Hits

Evacuate the Dancefloor (Radio Edit)
Cascada - Evacuate the Dancefloor

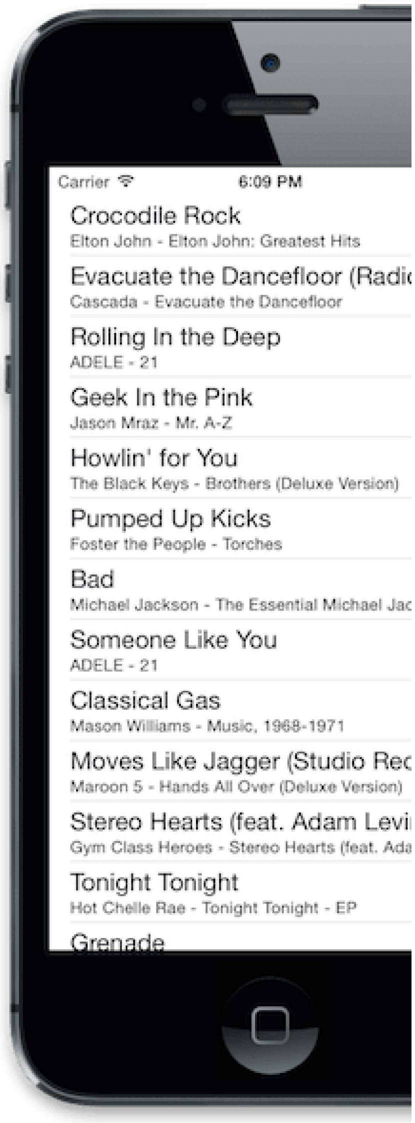
Rolling In the Deep
ADELE - 21

Geek In the Pink
Jason Mraz - Mr. A-Z

Howlin' for You
The Black Keys - Brothers (Deluxe Version)

Pumped Up Kicks
Foster the People - Torches

Bad
Michael Jackson - The Essential Michael



Crocodile Rock
Elton John - Elton John: Greatest Hits

Evacuate the Dancefloor (Radio
Cascada - Evacuate the Dancefloor

Rolling In the Deep
ADELE - 21

Geek In the Pink
Jason Mraz - Mr. A-Z

Howlin' for You
The Black Keys - Brothers (Deluxe Version)

Pumped Up Kicks
Foster the People - Torches

Bad
Michael Jackson - The Essential Michael Jac

Someone Like You
ADELE - 21

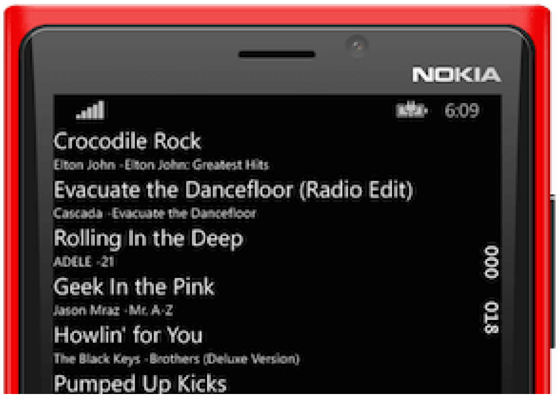
Classical Gas
Mason Williams - Music, 1968-1971

Moves Like Jagger (Studio Rec
Maroon 5 - Hands All Over (Deluxe Version)

Stereo Hearts (feat. Adam Levi
Gym Class Heroes - Stereo Hearts (feat. Ada

Tonight Tonight
Hot Chelle Rae - Tonight Tonight - EP

Grenade



Crocodile Rock
Elton John -Elton John: Greatest Hits

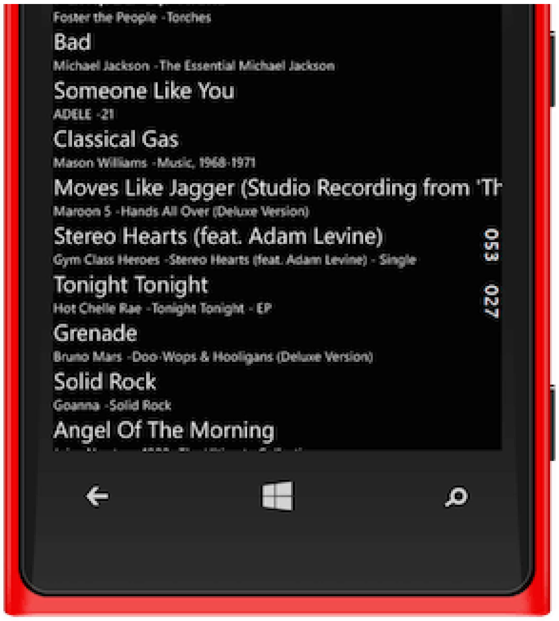
Evacuate the Dancefloor (Radio Edit)
Cascada -Evacuate the Dancefloor

Rolling In the Deep
ADELE -21

Geek In the Pink
Jason Mraz -Mr. A-Z

Howlin' for You
The Black Keys -Brothers (Deluxe Version)

Pumped Up Kicks



Foster the People - Torches

Bad

Michael Jackson - The Essential Michael Jackson

Someone Like You

ADELE - 21

Classical Gas

Mason Williams - Music, 1968-1971

Moves Like Jagger (Studio Recording from 'Th

Maroon 5 - Hands All Over (Deluxe Version)

Stereo Hearts (feat. Adam Levine)

Gym Class Heroes - Stereo Hearts (feat. Adam Levine) - Single

Tonight Tonight

Hot Chelle Rae - Tonight Tonight - EP

Grenade

Bruno Mars - Doo-Wops & Hooligans (Deluxe Version)

Solid Rock

Goanna - Solid Rock

Angel Of The Morning

Summary

In this exercise, we have created a Shared Project and used it in an iOS, Android and Windows Phone application. The library had both platform-agnostic code, as well as some platform-specific code isolated with conditional compilation.

Go Back