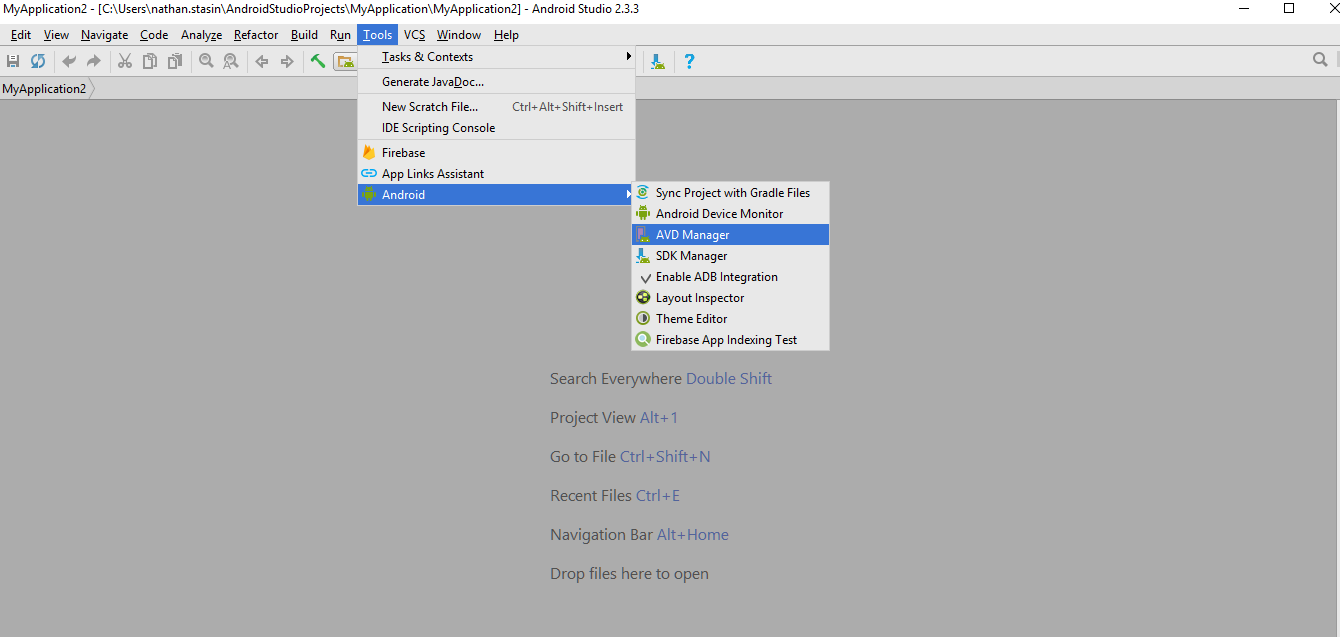
Xamarin

# Android Studio

Android Studio is the official [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) for [Google](https://en.wikipedia.org/wiki/Google)'s [Android](https://en.wikipedia.org/wiki/Android_(operating_system)) [operating system](https://en.wikipedia.org/wiki/Operating_system), designed specifically for [Android development](https://en.wikipedia.org/wiki/Android_software_development).

Your emulator must be running before visual studio can deploy the app to the emulator. Find the emulator:



# Xamarin vs. Xamarin.Forms

The [Xamarin SDK](http://open.xamarin.com/) provides bindings to the platform-specific APIs on each mobile platform, so you can call Android or iOS APIs from C# code. This allows you to build native apps using C#, but you still need to design the UI separately for each platform.

[Xamarin.Forms](https://www.xamarin.com/forms) is an additional layer on top of the Xamarin SDK that makes it possible to build your UI once (in XAML markup) and let Xamarin do the hard work of translating it into the appropriate UI elements on the target platform. You can drop down to the Xamarin SDK level and interact with the platform APIs if you need to.

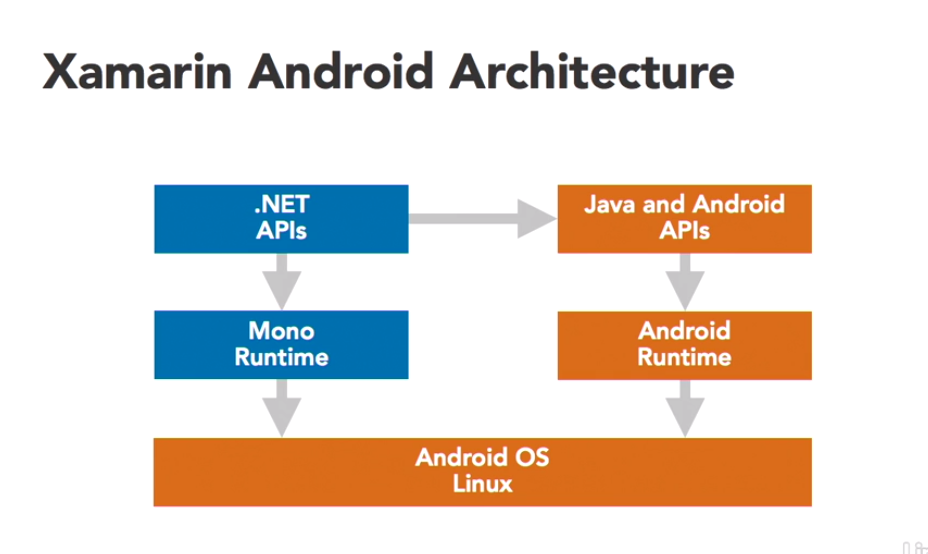
Xamarin forms have the idea of pages, which represent app screens (activities on Android). Controls are the UI elements shown on a page (eg buttons).

Should you use “raw” Xamarin, or Xamarin.Forms? It depends on what you are building:

* If you’re building an app that needs little platform-specific functionality or custom UI, go with Xamarin.Forms. This is a good choice for straightforward data-entry apps and prototypes.
* If you’re building an app that needs UI customized for each platform, or includes a lot of complex interactions, you’re better off with straight Xamarin.

# Xamarin

Its best to build the app as soon as you have created the solution, because certain things only happen on build (eg: Resource.Id.FirstButton)



Xamarin works by adding a second runtime to the phone, called the Mono runtime. Mono then communicates with the Android OS. Since most of the device specific features are not available to the Mono runtime, Xamarin provides bindings which can translate c# code to Java and Android APIs.

 An Activity is a special Android class which corresponds to a single application screen, and it is responsible for drawing and powering the user interface.

Android calls the Activity's OnCreate method when it creates the Activity (before the screen is presented to the user).

There are 7 lifecycle processes in an Android activity. They include −

* onCreate − It is called when the activity is first created.
* onStart − It is called when the activity starts and becomes visible to the user.
* onResume − It is called when the activity starts interacting with the user. User input takes place at this stage.
* onPause − It is called when the activity runs in the background but has not yet been killed.
* onStop − It is called when the activity is no longer visible to the user.
* onRestart − It is called after the activity has stopped, before starting again. It is normally called when a user goes back to a previous activity that had been stopped.
* onDestroy − This is the final call before the activity is removed from the memory.

**Portable Class Libraries** (or **PCL**s) attempt to address this problem by creating a library which is capable of running on multiple .NET platforms (Xamarin.iOS, Xamarin.Android, Windows, etc.). They provide a more structured container for sharing code which is not tied to a specific runtime but also require more architecture and thought when using platform-specific features.

android:text="@string/Hello"

android:id="@+id/button"

# .axml vs .xml

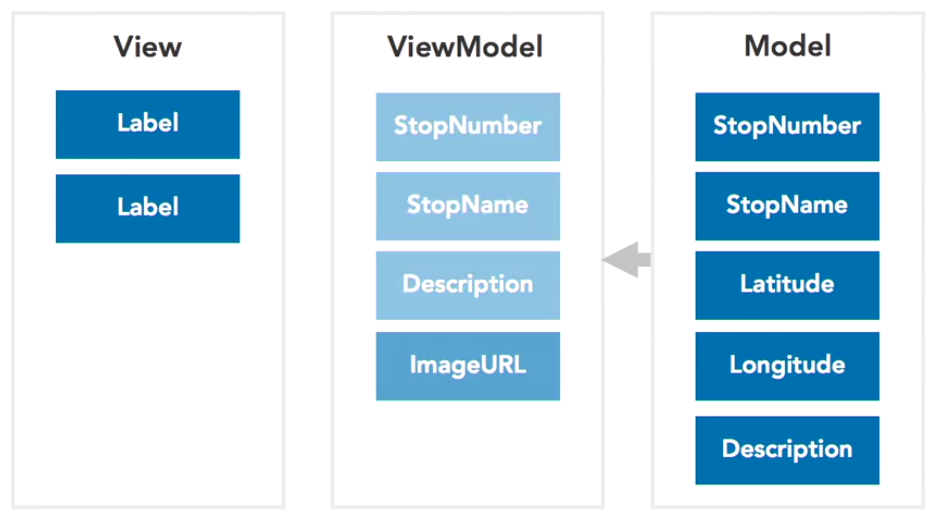
Xamarin uses axml, but the visual studio UI designer is a bit kak. Android studio uses .xml, but simply renaming the file extension doesn’t work. So, install the Xamaridea extension via Tools>>ExtensionsAndUpdates. You can then simply right click on the xml file in visual studio and select: Open in Android studio.

# DataBinding

Data binding connects two objects, called the source and the target. The source contains the data, the target is the destination for the data (eg a label). Can have one way or two way bindings. For example, in a two way binding, the user can change the text, which is then stored in the source object. Bindings use the binding engine built into Xamarin.

# MvvmCross

MVVM – Model, View, ViewModel. Similar to MVC for web development.



Deployed MvvmCross applications consist of two parts:

* one shared ‘core’ Portable Class Library (PCL) project, containing as much code as possible: models, view models, services, converters, etc
* one UI project per platform, each containing the bootstrap and view-specific code for that platform