Introduction to Mobile Development

Building mobile applications can be as easy as opening up your IDE, throwing something together, doing a quick bit of testing, and submitting to an App Store – all done in an afternoon. Or it can be an extremely involved process that involves rigorous up-front design, usability testing, QA testing on thousands of devices, a full beta lifecycle, and then deployment a number of different ways.

In this document, we're going to take a thorough introductory examination of building a Xamarin mobile application, including:

1. **Requirements** – We'll enumerate and examine the requirements for building for iOS and Android applications.
2. **Introduction to Xamarin** – List of features of the Xamarin platform.
3. **How Does Xamarin Work?** – A brief overview of how Xamarin works to bring C# to iOS and Android.
4. **Get started!** – Dive in to building your first Xamarin app, for iOS, Android, or all the platforms using Xamarin.Forms.

This document is intended to introduce the Xamarin platform. To learn more about the *process* of building mobile applications from design through to testing, refer to the Introduction to the Mobile Software Development Lifecycle document.

Requirements

If you'd like to develop for iOS, whether you want to code in Xamarin Studio or Visual Studio, you must have an Apple Macintosh computer running at least OS X Yosemite on hand. Although Xamarin applications are based on the .NET BCL and are written in C#, Xamarin.iOS requires the iOS SDK and Xcode in order to compile. Additionally, the iOS Device Simulator is part of the iOS SDK, and therefore only available on Mac. In order to download the iOS SDK, you must install Xcode, which can be downloaded for free from the

App Store. In addition to the iOS SDK, Xcode will also install iOS simulator for development and testing, and with the addition of free provisioning in Xcode 7, it is possible to test on device without being part of the Apple Developer program.

Note that in order to distribute your Xamarin.iOS you must be a member of Apple's Developer Program. This program requires an upgraded membership that costs $99 USD/year.

All the tutorials presented here are based on the latest version. Installation is covered in the next tutorial.

Introduction to Xamarin

When considering how to build iOS and Android applications, many people think that the native languages, Objective-C, Swift, and Java, are the only choice. However, over the past few years, an entire new ecosystem of platforms for building mobile applications has emerged.

Xamarin is unique in this space by offering a single language – C#, class library, and runtime that works across all three mobile platforms of iOS, Android, and Windows Phone (Windows Phone's native language is already C#), while still compiling native (non-interpreted) applications that are performant enough even for demanding games.

Each of these platforms has a different feature set and each varies in its ability to write native applications – that is, applications that compile down to native code and that interop fluently with the underlying Java subsystem. For example, some platforms only allow you to build apps in HTML and JavaScript, whereas some are very low-level and only allow C/C++ code. Some platforms don't even utilize the native control toolkit.

Xamarin is unique in that it combines all of the power of the native platforms and adds a number of powerful features of its own, including:

1. **Complete Binding for the underlying SDKs** – Xamarin contains bindings for nearly the entire underlying platform SDKs in both iOS and Android. Additionally, these

bindings are strongly-typed, which means that they're easy to navigate and use, and provide robust compile-time type checking and during development. This leads to fewer runtime errors and higher quality applications.

2. **Objective-C, Java, C, and C++ Interop** – Xamarin provides facilities for directly invoking Objective-C, Java, C, and C++ libraries, giving you the power to use a wide array of 3rd party code that has already been created. This lets you take advantage of existing iOS and Android libraries written in Objective-C, Java or C/C++. Additionally, Xamarin offers binding projects that allow you to easily bind native Objective-C and Java libraries using a declarative syntax.
3. **Modern Language Constructs** – Xamarin applications are written in C#, a modern language that includes significant improvements over Objective-C and Java such as *Dynamic Language Features* , *Functional Constructs* such as *Lambdas* , *LINQ* , *Parallel Programming* features, sophisticated *Generics* , and more.
4. **Amazing Base Class Library (BCL)** – Xamarin applications use the .NET BCL, a massive collection of classes that have comprehensive and streamlined features such as powerful XML, Database, Serialization, IO, String, and Networking support, just to name a few. Additionally, existing C# code can be compiled for use in your applications, which provides access to thousands upon thousands of libraries that will let you do things that aren't already covered in the BCL.
5. **Modern Integrated Development Environment (IDE)** – Xamarin uses Xamarin Studio on Mac OS X, and also Xamarin Studio or Visual Studio on Windows. These are both modern IDE's that include features such as code auto completion, a sophisticated Project and Solution management system, a comprehensive project template library, integrated source control, and many others.
6. **Mobile Cross Platform Support** – Xamarin offers sophisticated cross-platform support for the three major mobile platforms of iOS, Android, and Windows Phone. Applications can be written to share up to 90% of their code, and our Xamarin.Mobile library offers a unified API to access common resources across all three platforms. This can significantly reduce both development costs and time to market for mobile developers that target the three most popular mobile platforms.

Because of Xamarin's powerful and comprehensive feature set, it fills a void for application developers that want to use a modern language and platform to develop cross-platform mobile applications.

Note:

This Getting Started series focuses on getting started building iOS and Android applications. If you're interested in building for Windows Phone, Microsoft offers tutorials here. If you're interested in learning more about cross-platform development with Xamarin (including Windows Phone), you can find our guide here.

Let's take a look at how this all works.

How Does Xamarin Work?

Xamarin offers two commercial products: Xamarin.iOS and Xamarin.Android. They're both built on top of *Mono*, an open-source version of the .NET Framework based on the published .NET ECMA standards. Mono has been around almost as long as the .NET framework itself, and runs on nearly every imaginable platform including Linux, Unix, FreeBSD, and Mac OS X.

On iOS, Xamarin's *Ahead-of-Time* (*AOT*) Compiler compiles Xamarin.iOS applications directly to native ARM assembly code. On Android, Xamarin's compiler compiles down to *Intermediate Language* (*IL*), which is then *Just-in-Time* (*JIT*) compiled to native assembly when the application launches.

In both cases, Xamarin applications utilize a runtime that automatically handles things such as memory allocation, garbage collection, underlying platform interop, etc.

MonoTouch.dll and Mono.Android.dll

Xamarin applications are built against a subset of the .NET BCL known as the Xamarin Mobile Profile. This profile has been created specifically for mobile applications and packaged in the MonoTouch.dll and Mono.Android.dll (for iOS and Android respectively). This is much like the way Silverlight (and Moonlight) applications are built against the Silverlight/Moonlight .NET Profile. In fact, the Xamarin Mobile profile is equivalent to the

Silverlight 4.0 profile with a bunch of BCL classes added back in.

For a full list of available assemblies and classes, see the Xamarin.iOS Assembly List and the Xamarin.Android Assembly List

In addition to the BCL, these .dlls include wrappers for nearly the entire iOS SDK and Android SDK that allow you to invoke the underlying SDK APIs directly from C#.

Application Output

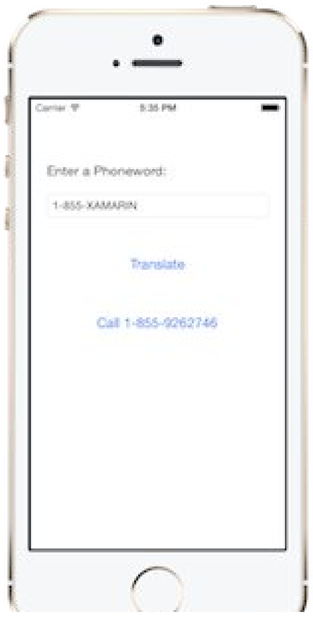
When Xamarin applications are compiled, the result is an Application Package, either an .app file in iOS, or .apk file in Android. These files are indistinguishable from application packages built with the platform's default IDEs and are deployable in the exact same way.

Getting Started

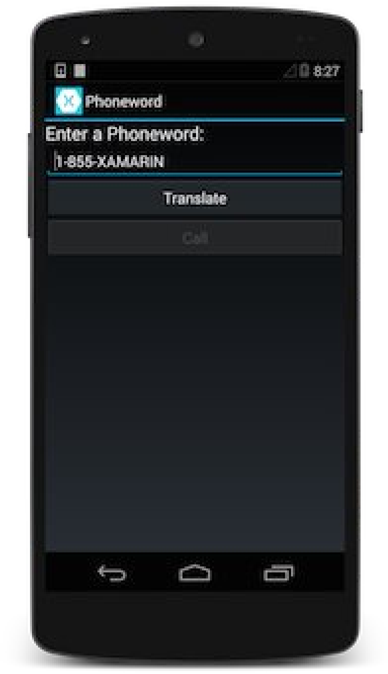
Now you've learned a little about how Xamarin works, it's time to dive in!

The next step is to start building your first app using one of these guides:

- **Hello, iOS**



- **Hello, Android**



- **Introduction to Xamarin.Forms**

Summary

This document has merely introduced the Xamarin platform. The real fun starts when you get your first app up-and-running. Check out the Hello, iOS, Hello, Android, and Introduction to Xamarin.Forms guides to begin.