Xamarin Cross-platform Application Development
Second Edition

Develop production-ready applications for iOS and Android using Xamarin

Jonathan Peppers
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 ‘Setting Up Xamarin’
- A synopsis of the book’s content
- More information on Xamarin Cross-platform Application Development Second Edition

About the Author

Jonathan Peppers is a Xamarin MVP and the lead developer of the popular apps and games at Hitcents, such as Hanx Writer and Draw a Stickman: EPIC. He has been working with the C# programming language for over 8 years. He is also quite familiar with other technologies, such as WinForms, WPF, Unity, ASP.Net WebForms, ASP .Net MVC, and Windows Azure. In recent years, Hitcents has been heavily investing in mobile development with Xamarin and has developed over 50 mobile applications with Xamarin across multiple platforms.
Xamarin has built three core products for developing iOS and Android applications in C#: Xamarin Studio, Xamarin.iOS, and Xamarin.Android. Xamarin gives you direct access to the native APIs on each platform and the flexibility to share C# code between platforms. Using Xamarin and C#, you get better productivity when compared to Java or Objective-C, and still retain great performance compared to an HTML or a JavaScript solution.

In this book, we will develop a real-world sample application to demonstrate what you can do with Xamarin technologies, and build on core platform concepts for iOS and Android. We will also cover advanced topics such as push notifications, retrieving contacts, using the camera, and GPS location. With Xamarin 3, a new framework was introduced called Xamarin.Forms. We will cover the basics of Xamarin.Forms and how you can apply it to cross-platform development. Finally, we will walk through what it takes to submit your application to the Apple App Store and Google Play.

What This Book Covers

Chapter 1, Setting Up Xamarin, covers the process of installing the appropriate Xamarin software and native SDKs required for performing cross-platform development.

Chapter 2, Hello, Platforms!, walks you through creating your first "Hello World" application on iOS and Android, which also covers some basic concepts on each platform.

Chapter 3, Code Sharing between iOS and Android, gives you code-sharing techniques and project setup strategies that can be used with Xamarin.

Chapter 4, XamChat – a Cross-platform App, introduces a sample application that we will be building throughout the book. In this chapter, we will write all the shared code for the application, complete with unit tests.

Chapter 5, XamChat for iOS, covers the technique of implementing the iOS user interface for XamChat and various iOS development concepts.

Chapter 6, XamChat for Android, covers the technique of implementing the Android version of XamChat and introduces Android-specific development concepts.
Chapter 7, *Deploying and Testing on Devices*, walks you through the painful process of deploying your first application to a device. We also cover why it is important to always test your application on real devices.


Chapter 9, *Third-party Libraries*, covers the various options of using third-party libraries with Xamarin and how you can even leverage native Java and Objective-C libraries.

Chapter 10, *Contacts, Camera, and Location*, introduces the library, Xamarin.Mobile, as a cross-platform way to access users' contacts, camera, and GPS location.

Chapter 11, *Xamarin.Forms*, discovers Xamarin's latest framework, Xamarin.Forms, and how you can leverage it to build cross-platform applications.

Chapter 12, *App Store Submission*, explains the process of submitting your app to the Apple App Store and Google Play.
Xamarin's development tools have given us the power to develop native iOS, Android, and Mac applications in C#, which is one of the most popular programming languages. There are many advantages of choosing Xamarin to develop mobile applications instead of Java and Objective-C. You can share code between both the platforms and can be more productive by taking advantage of the advanced language features of C# and the .NET base class libraries. Alternatively, you would have to write the app twice for Android and iOS and lose the benefits of garbage collection when using Objective-C.

In comparison to other techniques of developing cross-platform applications with JavaScript and HTML, Xamarin also has some distinct advantages. C# is generally more performant than JavaScript, and Xamarin gives developers direct access to the native APIs on each platform. This allows Xamarin applications to have a native look and perform in a manner similar to their Java or Objective-C counterparts.

Xamarin's tooling works by compiling your C# into a native ARM executable that can be packaged as an iOS or Android application. It bundles a stripped-down version of the Mono runtime with your application that only includes the features of the base class libraries your app uses.
In this chapter, we'll set up everything you need to get started on developing with Xamarin. By the end of this chapter, we'll have all the proper SDKs and tools installed and all the developer accounts needed for app store submission.

In this chapter, we will cover:

- An introduction to Xamarin tools and technology
- Installing Xcode, Apple's IDE
- Setting up all Xamarin tools and software
- Setting up the Android emulator
- Enrolling in the iOS Developer Program
- Registering for Google Play

The Xamarin tools

Xamarin has developed three core products for developing cross-platform applications: Xamarin Studio (formerly MonoDevelop), Xamarin.iOS (formerly MonoTouch), and Xamarin.Android (formerly Mono for Android). These tools allow developers to leverage the native libraries on iOS and Android and are built on the Mono runtime.

Mono, an open source implementation of C# and the .NET framework, was originally developed by Novell to be used on Linux operating systems. Since iOS and Android are similarly based on Linux, Novell was able to develop MonoTouch and Mono for Android as products to target the new mobile platforms. Shortly after their release, another company acquired Novell, and the Mono team left to form a new company. Very shortly after, Xamarin was founded to focus completely on these tools for developing with C# on iOS and Android.

Getting a development machine ready for cross-platform application development can take some time. And to make matters worse, Apple and Google both have their own requirements for development on their respective platforms. Let's go over what needs to be installed on your machine.
To get started on iOS, we'll need to install the following:

- **Xcode**: This is the core IDE for developing iOS and Mac applications in Objective-C
- **Xcode Command Line Tools**: These are installed inside Xcode, and provide common command-line tools and scripting languages that developers will find useful, such as Subversion, Git, Perl, and Ruby
- **The Mono runtime for Mac**: This is required for compiling and running C# programs on OS X
- **Xamarin.iOS**: This is Xamarin's core product for iOS development

Android also requires the following software to be installed to get started:

- **Java**: This is the core runtime for running Java applications on OS X
- **Android SDK**: This contains Google's standard SDK, device drivers, and emulators for native Android development
- **The Mono runtime for Mac**: This is required for compiling and running C# programs on OS X
- **Xamarin.Android**: This is Xamarin's core product for Android development

Each of these will take some time to download and install. If you can access a fast Internet connection, it will help speed up the installation and setup process. With everything ready to go, let's move ahead step-by-step, and hopefully, we can skip a few dead-ends you might otherwise run into.

It is important to note that Xamarin can also be used on Windows and Visual Studio, even though it is not covered in this book. A Mac is required for iOS development, so Windows developers must connect Visual Studio to a Mac to compile for iOS. Luckily, most of what we learn in this book can be directly applied to using Xamarin on Windows.
Installing Xcode

To make things progress more smoothly, let's start off by installing Xcode for Mac. Along with Apple's IDE, it will also install the most commonly used developer tools on the Mac. Make sure you have at least OS X 10.8 (Mountain Lion), and locate Xcode in the App Store, as shown in the following screenshot:

This will take quite some time to download and install. I'd recommend that you take the time to enjoy a nice cup of coffee or work on another project to pass the time.

When that is out of the way, launch Xcode for the first time and progress through the initial startup dialog. Next, navigate to Xcode | Preferences... to open Xcode's main settings dialog.

In the Downloads tab, you'll notice several additional packages you can install inside Xcode. Here, you can download the official iOS documentation, which the Xamarin installer will make use of. Optionally, you can install older iOS simulators, but we can just use the default one for the content in this book. When you're finished, your Xcode's Components section should look something similar to the following screenshot:
Installing Xcode installs the iOS SDK, which is a requirement for iOS development in general. As a restriction from Apple, the iOS SDK can only run on a Mac. Xamarin has done everything possible to make sure they follow Apple's guidelines for iOS, such as restricting dynamic code generation. Xamarin's tools also leverage features of Xcode wherever possible to avoid reinventing the wheel.

**Installing Xamarin**

After installing Xcode, there are several other dependencies that need to be installed in order prior to developing with Xamarin's tools. Luckily, Xamarin has improved the experience by creating a neat all-in-one installer.

Install the free Xamarin Starter Edition by performing the following steps:

1. Go to [http://Xamarin.com](http://Xamarin.com) and click on the large Download now button.
2. Fill out some basic information about yourself.
3. Download the XamarinInstaller.dmg file and mount the disk image.
4. Launch Install Xamarin.app and accept any OS X security warnings that appear.
Setting Up Xamarin

5. Progress through the installer; the default options will work fine. You can optionally install Xamarin.Mac, but this topic is not covered in this book.

The Xamarin installer will download and install prerequisites such as the Mono runtime, Java, the Android SDK (including the Android emulator and tools), and everything else you need to get up and running.

You will end up with something similar to what is shown in the following screenshot, and we can move on to conquer bigger topics in cross-platform development:

![Xamarin Installer Screenshot]

Choosing a Xamarin license

Xamarin's tools can seem a bit pricy to the casual observer, but I tend to think of it as how much time you will save using a more productive language such as C#. Additionally, their products will save you a good percentage of development time by enabling you to develop a cross-platform application instead of writing it twice in Java and Objective-C.

Xamarin has several editions, so it is good to know the differences in order to determine which license you might need to purchase. The editions are as follows:

- **Starter Edition**: This is available to individuals only, and it has a limit of 64 KB of compiled user code. Certain features are unavailable such as the Xamarin.Forms framework and calling into third-party native libraries.
• **Indie Edition**: This is available to individuals only, and it does not include Visual Studio support.

• **Business Edition**: This is available for companies; it adds features for Visual Studio and includes better Xamarin product support.

• **Enterprise Edition**: This includes prime components in the Xamarin Component Store for free and many more Xamarin support options such as hotfixes and less than 24 hours response time to issues.

---

### Setting up the Android emulator

The Android emulator has historically been known to be sluggish compared to developing on a physical device. To help solve this issue, Google has produced a new x86 emulator that supports hardware acceleration on desktop computers. It isn’t installed by default in the **Android Virtual Device (AVD)** Manager, so let’s set that up.

The x86 Android emulator can be installed by performing the following steps:

1. Open Xamarin Studio.
2. Navigate to **Tools** | **Open Android SDK Manager**.
3. Scroll down to **Extras**; install **Intel x86 Emulator Accelerator (HAXM)**.
   This will download an installer that we have to run.
4. Open Finder and press **Command + Shift + G** to open the navigation popup.
5. Navigate to `~/Library/Developer/Xamarin/android-sdk-macosx/extras/intel` and install the appropriate package (based on your Mac OS X version).
6. Scroll to **Android 4.4.2 (API 19)**; install **Intel x86 Atom System Image**.
7. Optionally, install any other packages you are interested in. As a shortcut, the Android SDK Manager automatically selects certain packages for you to install by default.
8. Close the Android SDK Manager and switch back to Xamarin Studio.
9. Navigate to **Tools** | **Open Android Emulator Manager**.
10. Click on **Create**.
11. Enter an AVD name of your choice, such as `x86 Emulator`.
12. Pick a generic device that will be appropriately sized for your display, such as one with a 4" WVGA display.
13. As **Target**, make sure that you select **Intel x86 Atom System Image**.

14. After creating the device, go ahead and click on **Start…** to make sure the emulator runs properly.

The emulator will take some time to start up, so it is a good idea to leave the emulator running while performing Android development. Xamarin is using the standard Android tools here, so you would have the same issue while developing with Java. If everything starts properly, you will see an Android boot screen followed by a virtual Android device ready for deploying applications from Xamarin Studio, as shown in the following screenshot:
Enrolling in the iOS Developer Program

To deploy to an iOS device, Apple requires membership to its iOS Developer Program. Membership is $99 USD per year and gives you access to deploy 200 devices for development purposes. You also get access to test servers for implementing more advanced iOS features such as in-app purchases, push notifications, and iOS Game Center. Testing your Xamarin.iOS applications on a physical device is important, so I recommend that you get an account prior to starting iOS development. Performance is very different in a simulator running on your desktop versus a real mobile device. There are also a few Xamarin-specific optimizations that only occur when running on a real device. We'll fully cover the reasons for testing your apps on devices in the later chapters.

Signing up for the iOS Developer Program can be performed through the following steps:

2. Click on **Enroll Now**.
3. Sign in with an existing iTunes account or create a new one. This can't be changed later, so choose one that is appropriate for your company.
4. Enroll either as an individual or a company. Both are priced at $99; but, registering as a company will require paperwork to be faxed to Apple with the assistance of your company's accountant.
5. Review the developer agreement.
6. Fill out Apple's survey for developers.
7. Purchase the $99 developer registration.
8. Wait for a confirmation e-mail.
You should receive an e-mail that looks something similar to the following screenshot within two business days:

```
Thank You for Joining an Apple Developer Program

Dear Jonathan Peppers,

Thank you for joining the iOS Developer Program. You now have access to a comprehensive set of development tools and resources to assist you in developing innovative apps.

Please feel free to contact us if you have any general questions or require assistance.

Best regards,

Apple Developer Support
```

From here, we can continue setting up your account:

1. Either click on Log in now from the e-mail you received or go to https://itunesconnect.apple.com.
2. Log in with your earlier iTunes account.
3. Agree to any additional agreements that appear on the home page of your dashboard.
4. From the iTunes Connect dashboard, navigate to Agreements, Tax, and Banking.
5. In this section, you will see three columns for Contact Info, Bank Info, and Tax Info.
6. Fill out the appropriate information for your account in all of these sections. Assistance from an accountant will most likely be needed for a company account.
When all is said and done, your **Contracts, Tax, and Banking** section should look something similar to the following screenshot:

With your iOS developer account successfully registered, you will now be able to deploy to iOS devices and publish your apps to the Apple App Store.

### Registering as a Google Play developer

Unlike iOS, deploying your applications to Android devices is free and just requires a few changes in your device settings. A Google Play developer account has only a one-time fee of $25 and doesn't have to be renewed each year. However, just like iOS, you will need a Google Play account to develop in-app purchases, push notifications, or Google Play game services. I would recommend that you set up an account ahead of time if you inevitably plan on submitting an app to Google Play or need to implement one of these features.

To register as a developer for Google Play, perform the following steps:

2. Log in with an existing Google Account or create a new one. This can't be changed later, so choose one that is appropriate for your company if needed.
Setting Up Xamarin

3. Accept the agreement and enter your credit card information.
4. Choose a developer name and enter other important information for your account. Again, choose names appropriate for your company to be seen by users in the app store.

If everything is filled out correctly, you will end up with the following Google Play Developer Console:

If you plan on selling paid apps or in-app purchases, at this point, I would recommend that you set up your Google merchant account. This will enable Google to pay you the proceeds toward your app sales by applying the appropriate tax laws in your country. If you are setting this up for your company, I would recommend that you get the assistance of your company's accountant or bookkeeper.
The following are the steps to set up a Google merchant account:

1. Click on the **set up a merchant account** button.
2. Log in with your Google account a second time.
3. Fill out the appropriate information for selling apps: address, phone number, tax information, and a display name to appear on your customers' credit card bill.

When done, you will see that the help tip for setting up a merchant account is now missing from the developer console, as shown in the following screenshot:
At this point, one would think that our account would be fully set up, but there is one more crucial step prior to being able to sell apps: we have to enter the banking information.

Setting up banking for your Google merchant account can be performed with the following steps:

2. Click on the Financial Reports section.
3. Click on the small link titled Visit your merchant account for details.
4. You should see a warning indicating that you do not have a bank account set up. Click on the Specify a Bank Account link to get started.
5. Enter your banking information. Again, a company accountant might be needed.
6. In a few days, look for a small deposit in your account from Google.
8. Click on the Settings tab, then Financials.
9. Next, click on Verify Account.
10. Enter the amount that appeared on your bank account and click on Verify deposit.

Your Google merchant account is also the place where you can cancel or refund customer orders. Google Play is different from the iOS App Store in that all customer issues are directed to the developers.

Summary

In this chapter, we discussed Xamarin's core products for developing Android and iOS applications in C#: Xamarin Studio, Xamarin.iOS, and Xamarin.Android. We installed Xcode and then ran the Xamarin all-in-one installer, which installs Java, the Android SDK, Xamarin Studio, Xamarin.iOS, and Xamarin.Android. We set up the x86 Android emulator for a faster, more fluid experience when debugging applications. Finally, we set up iOS and Google Play developer accounts for distributing our applications.

In this chapter, you should have acquired everything you need to get started on building cross-platform applications with Xamarin. Your development computer should be ready to go and you should have all the native SDKs installed and ready for creating the next great app to take the world by storm.
The concepts in this chapter will set us up for more advanced topics that will require the proper software installed as well as developer accounts with Apple and Google. We will be deploying applications to real devices and implementing more advanced features such as push notifications. In the next chapter, we'll create our first iOS and Android application and cover the basics of each platform.
Where to buy this book

You can buy Xamarin Cross-platform Application Development Second Edition from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.