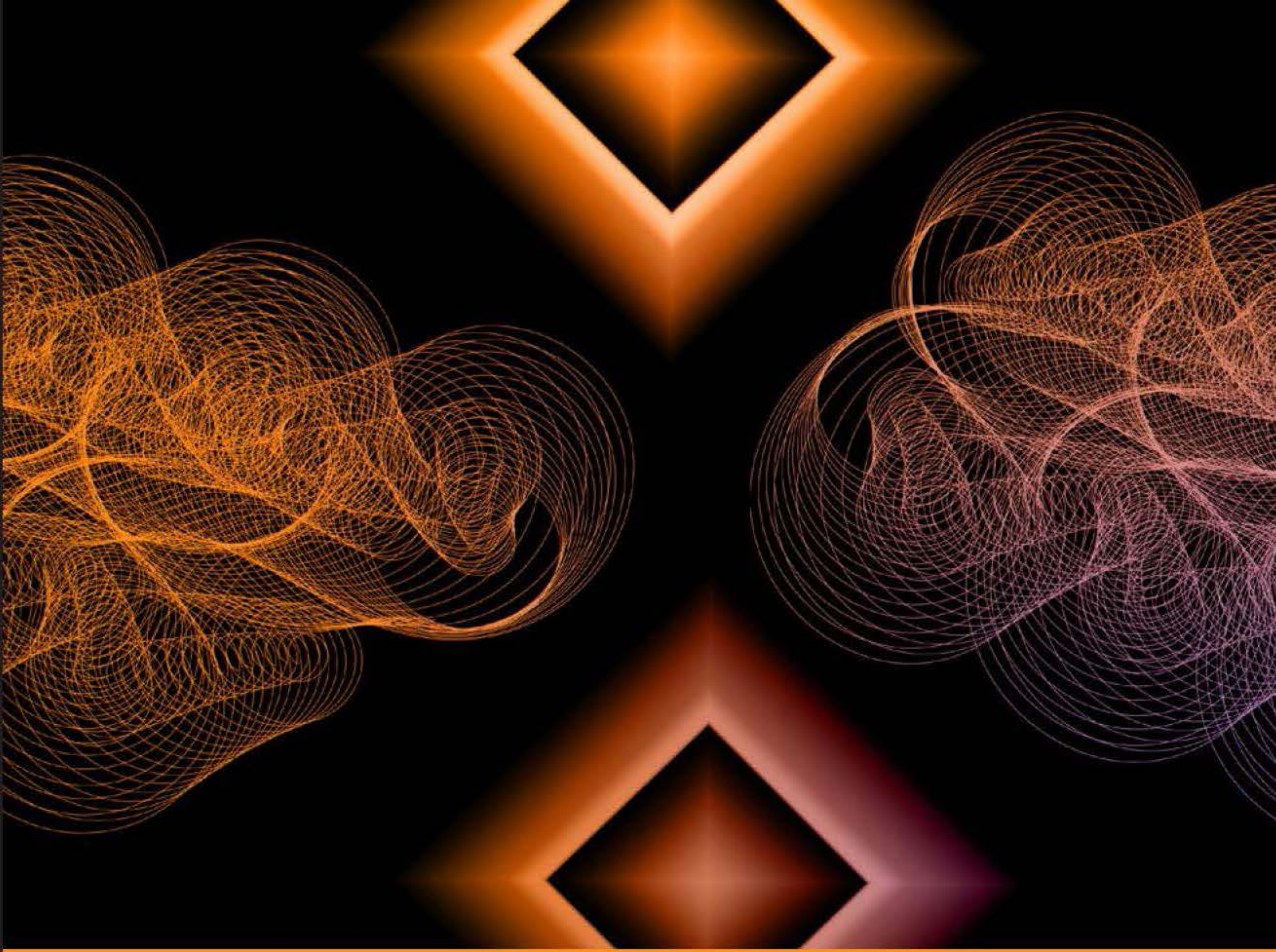


Xamarin Cross-platform Application Development

Develop production-ready applications for iOS and Android using Xamarin

Jonathan Peppers

[PACKT] PUBLISHING

Xamarin Cross-platform Application Development

Develop production-ready applications for iOS and Android using Xamarin

Jonathan Peppers

[PACKT]
PUBLISHING

BIRMINGHAM - MUMBAI

Xamarin Cross-platform Application Development

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: February 2014

Production Reference: 2110214

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84969-846-7

www.packtpub.com

Cover Image by Yuvraj Mannari (yuvrajm@packtpub.com)

Credits

Author
Jonathan Peppers

Reviewers
Pierce Boggan
Betim Drenica
Tom Opgenorth
William Thomas

Acquisition Editors
Sam Birch
Richard Harvey

Content Development Editor
Neeshma Ramakrishnan

Technical Editors
Tanuj Gulati
Adrian Raposo

Copy Editors
Dipti Kapadia
Kirti Pai

Project Coordinator
Kranti Berde

Proofreaders
Simran Bhogal
Ameesha Green

Indexers
Hemangini Bari
Tejal Soni

Graphics
Ronak Dhruv
Abhinash Sahu

Production Coordinator
Nilesh Bambardekar

Cover Work
Nilesh Bambardekar

About the Author

Jonathan Peppers is a Xamarin MVP and the lead developer of the popular cross-platform game, Draw a Stickman: EPIC. Jon works for Hitcents, a software development company based in Bowling Green, Kentucky. He has been working with the C# programming language for over 7 years. He has also been working with technologies such as WinForms, WPF, ASP.NET WebForms, and ASP.NET MVC. In recent years, Hitcents has been heavily investing in mobile development with Xamarin and has developed nearly 40 mobile applications across multiple platforms.

About the Reviewers

Pierce Boggan is a student of Software Engineering at the Auburn University. He has a passion for software, entrepreneurship, and theology. In the past, Pierce has worked for Xamarin as a Customer Support Engineering Intern. He also hosts a bi-monthly podcast with Chris Hardy in which all things related to Xamarin and mobile development are discussed. You can find him blogging about software, specifically mobile development, at `www.pierceboggan.com`.

I would like to thank my Lord and Savior Jesus Christ, as well as my close friends and family who have supported me along the way.

Betim Drenica is a professional developer with over a decade's experience and has specialized in many technologies from database to client technologies to mobiles. He considers himself as a software architect oriented in Service Oriented Applications (RESTing for a long time) paying more attention to mobile apps for all major platforms (Windows Phone, Android, and iOS) using Xamarin. Being a very community-driven person, he is the founder and leader of two user groups, Albanian SQL Server UG and Albanian .NET UG. He is a Microsoft Certified Trainer and a regular speaker at many tech conferences. He's a blogger and you can find him at `www.betimdrenica.wordpress.com` and usually tweets from time to time on his Twitter account `@betimdrenica`. He also spends a good amount of his time on LinkedIn at `http://www.linkedin.com/in/betim`.

Tom Opgenorth is an IT professional from Alberta, Canada. His interest in computing started in 1983 when he taught himself Applesoft Basic and steadily progressed from there on. For the past 20 years, he has taken on many roles in IT, such as developer/engineer, QA, team lead, and architect. Tom settled on the .NET platform in 2002 with an interest in C#, and became a fan of Mono in 2003. He was awarded the Microsoft MVP award for C# in 2008, 2009, and 2010. In early 2008, he purchased his first smartphone, the Android Developer Phone 1, and has been hooked on mobile technologies with a focus on Android ever since.

Tom is currently working for Xamarin. He spends his days with the awesome documentation team fusing C# and Android into Xamarin documentation with the goal of helping developers create cross-platform mobile applications. His evenings are spent taking his two boys around to their activities and being dragged around by his trusting Samoyed, who never says no to a walk.

William Thomas grew up in Claremont, CA, and developed an interest in computers from a young age. He started out with Flash development, became interested in web design, and then moved on to web development. After attending Cal Poly Pomona to study Computer Science, he began working in 2011, at Flavorus; the Los Angeles-based ticketing company.

Since his graduation, at Cal Poly, William continues to work full time at Flavorus, developing applications for web and mobile devices. He has most recently developed the Flavorus Box Office app for iPhone and iPad using the Xamarin .NET framework.

I'd like to acknowledge my friends and colleagues from Cal Poly Pomona and Flavorus. I truly appreciate the time spent with my professors in the classroom as well as the knowledge and experience I've gained at my workplace.

Table of Contents

Preface

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 among platforms. Using Xamarin and C#, the productivity you get is better than that of Java or Objective-C while maintaining a greater performance output compared to a HTML or 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 a camera, and GPS location. 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, *Xamarin Setup*, covers the process of installing the appropriate Xamarin software and native SDKs required for performing cross-platform development.

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

Chapter 3, *Code Sharing Between iOS and Android*, introduces code-sharing techniques and strategies to set up projects 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*, explains the painful process of deploying your first application on a device. We also cover why it is important to always test your application on real devices.

Chapter 8, *Web Services with Push Notifications*, explains the technique of implementing a real backend web service for XamChat using Azure Mobile Services.

Chapter 9, *Third-party Libraries*, covers the various options to use 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 library for accessing users' contacts, camera, and GPS location.

Chapter 11, *App Store Submission*, explains the process of submitting your app to the Apple App Store and Google Play.

What you need for this book

For this book, you will need a Mac computer that at least runs on OS X 10.7 Lion. Apple requires iOS applications to be compiled on a Mac, so our examples throughout the book will use Xamarin Studio on a Mac. You will also need a license of the business edition of Xamarin.Android and Xamarin.iOS. A free 30-day trial is also available. You can try the free Starter edition of Xamarin, but some of the more advanced examples will not work with that edition. Visit `http://xamarin.com/download` to download the appropriate software.

Who this book is for

If you are a developer who is already familiar with C# and you want to start mobile application development with Xamarin, this book is for you. If you have worked with ASP.NET, WPF, WinRT, or Windows Phone, then you will feel right at home using this book to develop native iOS and Android applications.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles are shown as follows: "Create a new folder in the `XamChat.Droid` project named `Core`."

A block of code is set as follows:

```
private async void LoadData()
{
  var service = ServiceContainer.Resolve<IWebService>()
    as AzureWebService;

  await service.LoadData();
}
```

Any command-line input or output is written as follows:

```
GoogleAnalytics.Droid: Warning BG8102:
    Class GoogleAnalytics.Tracking.CampaignTrackingService has
    unknown base type android.app.IntentService (BG8102)
    (GoogleAnalytics.Droid)
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "To get started, simply right-click on the folder, and select **Get More Components** to launch the store dialog."

[Warnings or important notes appear in a box like this.]

[Tips and tricks appear like this.]

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title via the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/submit-errata`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from `http://www.packtpub.com/support`.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
Xamarin Setup

Xamarin has finally given us the power to develop native iOS, Android, and Mac applications in C#, which is one of our favorite programming languages. There are many advantages of choosing Xamarin to develop mobile applications instead of Java and Objective-C. You can share code between multiple 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 in iOS.

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 the 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 let the Mono team go. 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 developing 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**: Apple's 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 would find useful, such as Subversion or SVN, Git, Perl, 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:

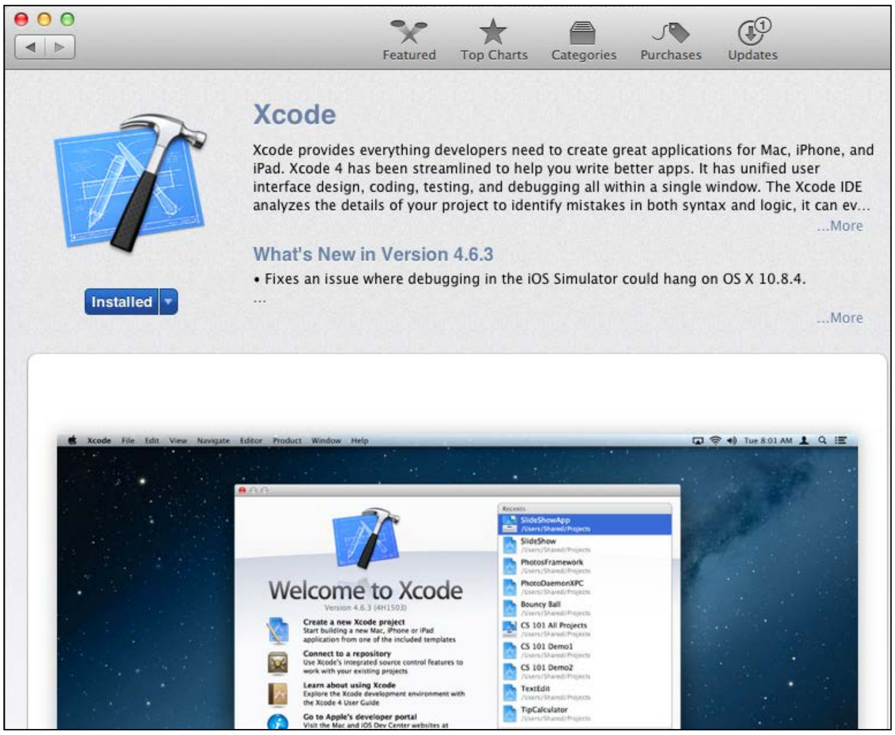
- **Java**: This is the core runtime for running Java applications on OS X
- **Android SDK**: This contains Google's standard SDK, device drivers, and emulators for native Android development
- **The Mono runtime for Mac**: This is required for compiling and running C# programs on OS X
- **Xamarin.Android**: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



Xcode

Xcode provides everything developers need to create great applications for Mac, iPhone, and iPad. Xcode 4 has been streamlined to help you write better apps. It has unified user interface design, coding, testing, and debugging all within a single window. The Xcode IDE analyzes the details of your project to identify mistakes in both syntax and logic, it can ev...

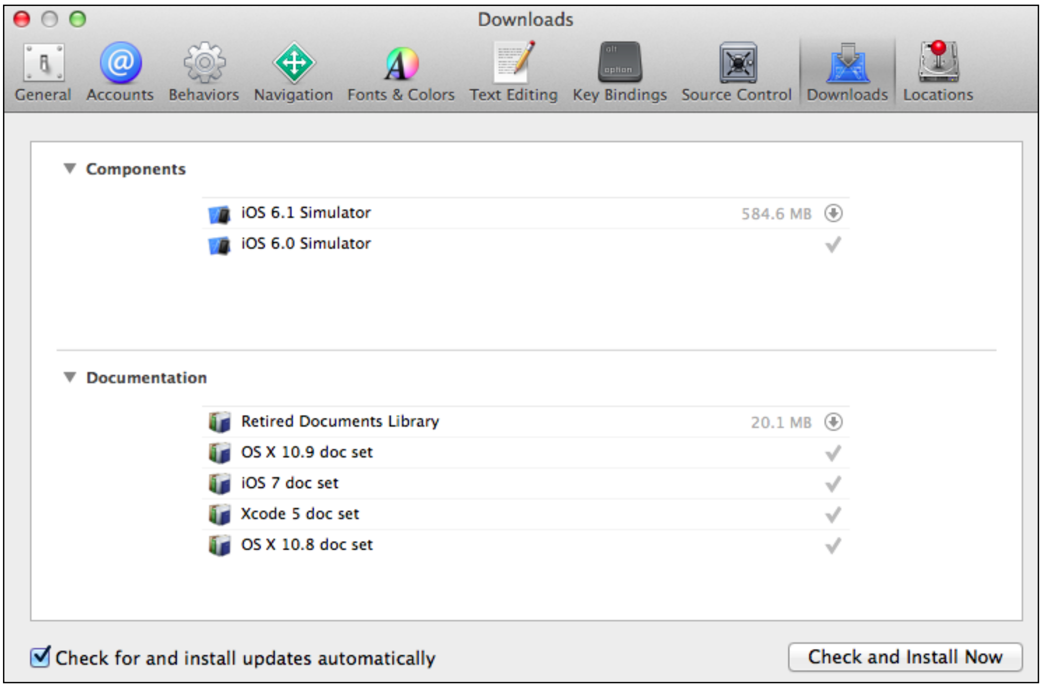
What's New in Version 4.6.3

• Fixes an issue where debugging in the iOS Simulator could hang on OS X 10.8.4.

This will take quite some time to download and install. I'd recommend taking the time to enjoy a nice cup of coffee or work on another project at the same 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** | **Components** tab, you'll notice several additional packages you can install inside Xcode. Go ahead and install the Command Line Tools. 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:



▼ Components

iOS 6.1 Simulator	584.6 MB	⊕
iOS 6.0 Simulator		✓

▼ Documentation

Retired Documents Library	20.1 MB	⊕
OS X 10.9 doc set		✓
iOS 7 doc set		✓
Xcode 5 doc set		✓
OS X 10.8 doc set		✓

☑ Check for and install updates automatically

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 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 to start developing with Xamarin's tools. Luckily, Xamarin has improved the experience by creating a neat all-in-one installer.

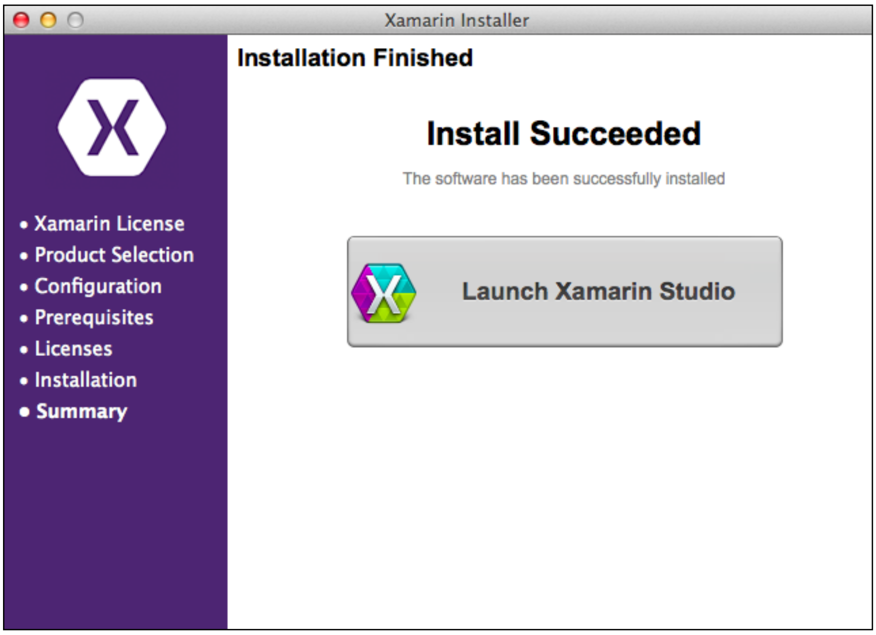
Install the free Xamarin Starter Edition with the following steps:

1. Go to `http://Xamarin.com` and click on the large **Download Now** button.

2. Fill out some basic information about yourself.
3. Download `XamarinInstaller.dmg` and mount the disk image.
4. Launch `Install Xamarin.app` and accept any OS X security warnings that appear.
5. Progress through the installer, the default options will work fine. You can optionally install Xamarin.Mac, but that 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 be 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:



Installation Finished

Install Succeeded

The software has been successfully installed

Xamarin has several editions, so it is good know the differences to determine which license you may 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
- **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 much more Xamarin product support such as hotfixes and less than 24 hours response time to issues

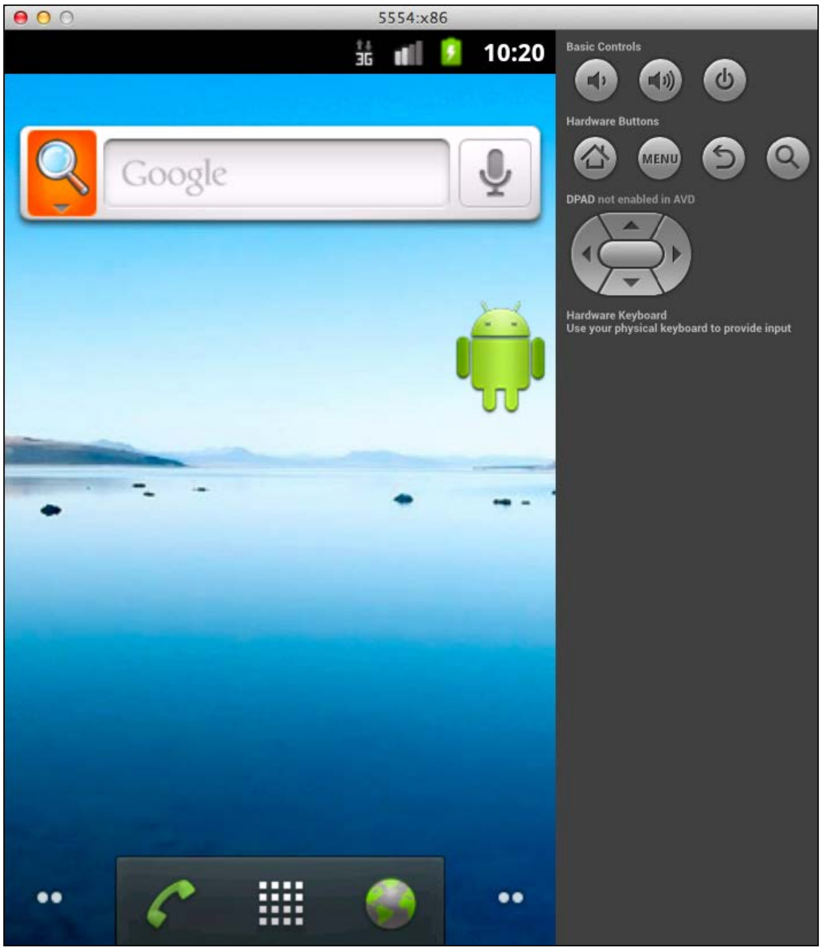
Setting up the Android emulator

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. Launch **Tools** | **Open Android SDK Manager…**.
3. Scroll down to **Extras**; install **Intel x86 Emulator Accelerator (HAXM)**.
4. Scroll to **Android 4.2.2 (API 17)**; install **Intel x86 Atom System Image**.
5. Optionally, install any other packages you are interested in. At a minimum, make sure you have everything that the Android SDK Manager selects for you to install by default.
6. Close the **Android SDK Manager** and switch back to Xamarin Studio.
7. Launch **Tools** | **Open AVD Manager…**.
8. Click on **New…**.
9. Enter an AVD name of your choice, such as `x86 Emulator`.
10. Pick a generic device that will be appropriately sized for your display, such as the **4.0 inch WVGA**.
11. As **Target**, make sure you select **Intel x86 Atom System Image**.
12. 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 performing Android development 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 a 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 getting 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 the device. We'll fully cover the reasons for testing your apps on devices in later chapters.