Beginning Xamarin Development for the Mac

Create iOS, watchOS, and Apple tvOS apps with Xamarin.iOS and Visual Studio for Mac

Dawid Borycki

Apress®

Beginning Xamarin Development for the Mac

Create iOS, watchOS, and Apple tvOS apps with Xamarin.iOS and Visual Studio for Mac

Dawid Borycki

Apress®

Beginning Xamarin Development for the Mac

Dawid Borycki
Institute of Physical Chemistry,
Polish Academy of Sciences,
Kasprzaka 44/52, Warsaw, 01-224, Poland

ISBN-13 (pbk): 978-1-4842-3131-9 ISBN-13 (electronic): 978-1-4842-3132-6
https://doi.org/10.1007/978-1-4842-3132-6

Library of Congress Control Number: 2017963095

Cover image designed by Freepik

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Joan Murray
Development Editor: Laura Berendson
Technical Reviewer: Chaim Krause
Coordinating Editor: Jill Balzano
Copy Editor: April Rondeau
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484231319. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

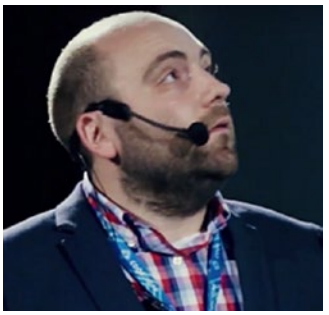
This book is dedicated to my wife, Agnieszka, daughter, Susanna, and son, Xavier, with love.

We all have dreams. But in order to make dreams come into reality, it takes an awful lot of determination, dedication, self-discipline, and effort.

—Jesse Owens

Contents

About the Author



Dawid Borycki is a software engineer, biomedical researcher, and an expert in several Microsoft developer technologies. He has resolved a broad range of software development challenges for device prototypes (mainly medical equipment), embedded device interfacing, and desktop and mobile programming. Dawid regularly speaks at international developers conferences and has published, cited, and presented on numerous developer topics, including web technologies, mobile/cross-platform development, wearables, embedded, and more.

About the Technical Reviewer

Chaim Krause is an expert computer programmer with over thirty years of experience to prove it. He worked as a lead tech support engineer for ISPs as early as 1995 and as a senior developer support engineer with Borland for Delphi, and has worked in Silicon Valley for over a decade in various roles, including technical support engineer and developer support engineer. He is currently a military simulation specialist for the US Army's Command and General Staff College, working on projects such as developing serious games for use in training exercises. He has also authored several video training courses on Linux topics and has been a technical reviewer for over twenty books, including *iOS Code Testing*, *Android Apps for Absolute Beginners* (4th ed.), and *XML Essentials for C# and .NET Development* (all Apress). It seems only natural then that he would be an avid gamer and have his own electronics lab and server room in his basement. He currently resides in Leavenworth, Kansas, with his loving partner, Ivana, and a menagerie of four-legged companions: their two dogs, Dasher and Minnie, and their three cats, Pudems, Talyn, and Alaska.

Acknowledgments

Dear reader, you hold or view this book thanks to Joan Murray, who was very encouraging of my book proposal and provided a lot of writing tips.

I am grateful to Chaim Krause for providing a very detailed technical review and catching even the smallest mistakes.

Many thanks go to Laura Berendson for reading the manuscript, providing feedback, and giving general advice. I thank Jill Balzano for her patience and keeping track of the book project. I also thank Welmoed Spahr and Todd Green for publishing this book and April Rondeau for copyediting.

Finally, special thanks go to my wife, Agnieszka, daughter, Susanna, and son, Xavier, for their continuous support and patience shown to me during the writing of this book.

Introduction

The programming of mobile apps has recently become one of the most important and exciting aspects of the IT market, leading to numerous applications. At the same time, mobile development is very challenging because of market fragmentation, which is manifested by the presence of devices of different sizes, operating systems, and software development tools. So, to start doing mobile development, you need to take into account these various factors and decide your strategy.

There are three leading mobile platforms: Apple (including iOS, watchOS, and tvOS), Android, and Universal Windows Platform (UWP), each of which provides dedicated tools for developing apps. Each platform has specific hardware requirements. Clearly, iOS SDKs require you to use the local Mac machine or the remote build agent installed on a Mac. UWP apps can be built on Windows 10 machines. Finally, you can use either Mac or Windows machines to develop Android apps. So, first and foremost, the question to answer is that of which hardware to choose. For mobile development, the Intel-based Mac platform is the best choice because it allows you to use macOS and Windows 10 concurrently. The latter can be installed natively through the Boot Camp or virtually using the Parallels Desktop. Accordingly, Mac machines will give you the most flexibility.

After choosing the hardware, you need to select the software development strategy. You have three traditional options here:

- Native apps – In this case, you use platform SDKs, which require you to utilize platform-specific IDEs and programming languages. For instance, in terms of iOS, watchOS, and tvOS, you would need to use either Swift or Objective-C along with Xcode. Native tools give you full access to the platform API at the cost of needing to learn platform-specific programming languages.

- Hybrid apps – They are programmed with web technologies as web pages. These pages are then rendered with a native WebView component, which can be virtually understood as the local web browser, delivered by each platform. In this case, you have a lot of flexibility in terms of programming tools and can use the same code across various platforms at the cost of reduced access to platform-specific APIs.

- Mobile web apps – They are web apps whose views are tailored to mobile devices. Such a strategy is the simplest to apply but does not let you access device components, and it also requires a network connection.

Of course, development strategy is dictated by a number of factors, ranging from your programming preferences to application complexity and target platforms. If you intend to target multiple platforms at the same time, you can choose one of the cross-platform mobile tools, like Xamarin.Forms, Qt Mobile, Embarcadero RAD Studio, or React Native. They provide an additional layer of the unified, platform-independent programming API, which is translated to a platform-specific API during compilation. Importantly, cross-platform instruments provide not only access to the API but also to visual controls. For instance, if you create a message dialog, it will be converted during compilation to the platform-specific message dialog. As a result, in such a "Write once, run anywhere (WORA)" approach, you can indeed compile the same code across various platforms at the cost of flexibility. Finally, if your goal is to extensively utilize platform-specific APIs, you will choose Xamarin.iOS or Xamarin.Android. These let you develop

native apps with C# or F#. Xamarin.iOS and Xamarin.Android provide an additional thin layer that maps C# or F# code onto the native platform API. You typically use Xamarin.iOS and Xamarin.Android to implement logic and design the UI separately for each platform. Although you have the extra work of creating platform-specific UIs, you can still share the platform-independent code between various apps. Such an approach gives you two important advantages:

- Your app is native. So, it looks like any other built-in app, as it utilizes native visual controls.
- Your app is written in modern programming language. As a result, you have easy access to numerous libraries and a wide community. This is especially important when you are C# programmer who wants to start developing mobile apps.

In this book, we will learn how to use Xamarin.iOS to develop apps for Apple devices: iPhone and iPad (iOS), Apple Watch (watchOS), and Apple TV (tvOS). We will first prepare the development tools (Visual Studio for Mac) and will get to know available project templates (Chapter 1). Subsequently, I will explicitly show how Xamarin.iOS is related to the native SDKs delivered by Apple, and we will investigate the app structure and lifecycle (Chapter 2). Then, we will learn how to create views (Chapter 3) and implement navigation between them (Chapter 4). Afterward, we will work with touch gestures (Chapter 5), study how to achieve high-quality apps with automatic testing (Chapter 6), and consume data from RESTful web services (Chapter 7). Finally, we will learn how to develop apps for Apple Watches (Chapter 8) and Apple TVs (Chapter 9).

In this book, however, we will not learn how to reuse code between various platforms nor how to develop WORA apps with Xamarin.Forms. You can find more information about this in this book *Beginning Visual Studio for Mac: Build Cross-Platform Apps with Xamarin and .NET Core* by Alessandro Del Sole.

CHAPTER 1

Fundamentals

In this chapter, I will guide you through the installation of the development tools you will need for this book. Specifically, we will install Visual Studio for Mac as well as Xcode. The latter is the native toolset for Mac developers. It delivers IDE, SDKs, and also the device simulators we will use. If you have these tools already installed, you can skip to the subsequent section.

After ensuring that all tools are ready, we will create the first Xamarin.iOS app for the iPhone and iPad. This app, shown in Figure 1-1, displays various alerts and responds to user actions. I will also discuss the available project templates that are delivered by Visual Studio. The same templates are available in Xcode, so Xamarin.iOS and Visual Studio let you access iOS platform–specific programming interfaces in a way similar to native development tools but with the ease and smoothness provided by the C# programming language. In this chapter, I will also discuss the basic aspects of designing user interfaces in Visual Studio and show you how to associate event handlers with events fired by visual controls.

Note In this chapter, I will not discuss Visual Studio for Mac in detail. I will only discuss the necessary elements of this IDE. You can find a comprehensive description of Visual Studio for Mac in the book *Beginning Visual Studio for Mac. Build Cross-Platform Apps with Xamarin and .NET Core* by Alessandro Del Sole.

D. Borycki, *Beginning Xamarin Development for the Mac*, https://doi.org/10.1007/978-1-4842-3132-6_1

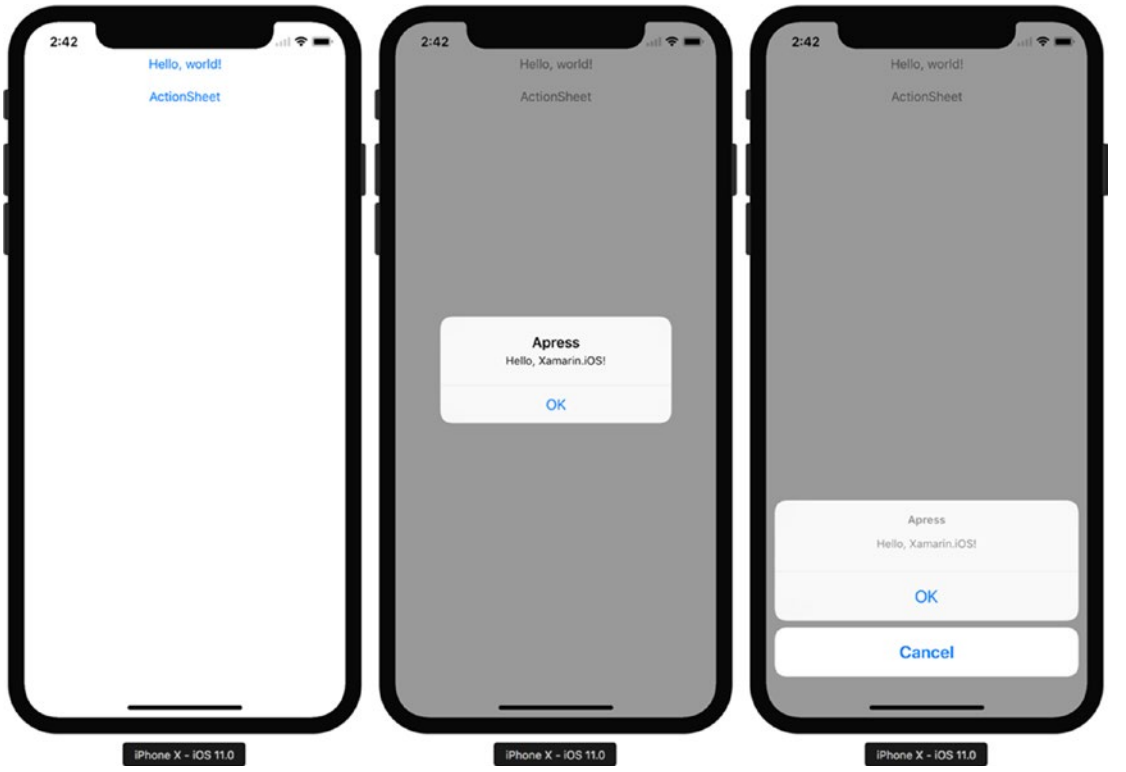


Figure 1-1. *The Hello, World! app we will build in this chapter. The app is executed in the iPhone X simulator.*

Setting Up the Development Environment

To install Visual Studio for Mac, you'll need a Mac with macOS Sierra 10.12 or above. Here, I'll be using either a MacBook Pro or iMac with macOS Sierra 10.16. Once you know that you meet basic platform requirements, you can download the Visual Studio installer from the following website: `http://bit.ly/vs-mac`. Once you have downloaded the installation package, run the installer. A window appears, as shown in Figure 1-2. In this window, you double-click the icon with a down arrow. Subsequently, you will see a dialog informing you that the installer was downloaded from the internet (Figure 1-3). Click the *Open* button to continue.

Figure 1-2. *An installer for Visual Studio for Mac*

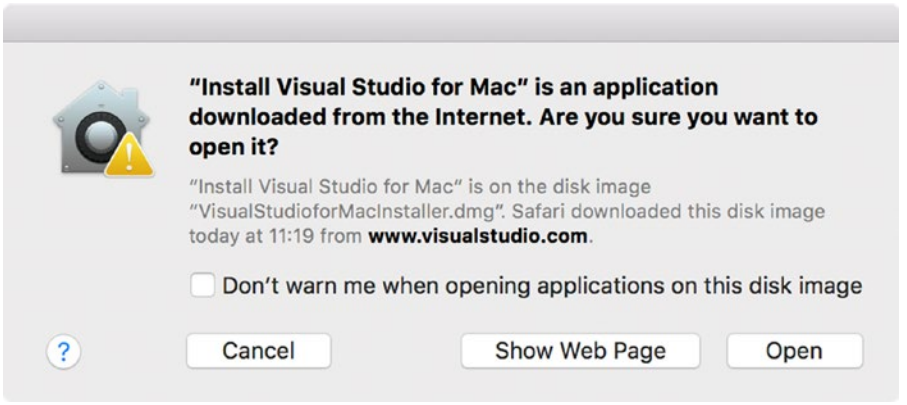


Figure 1-3. *A confirmation dialog*

Visual Studio installer will now verify your system configuration (Figure 1-4). More specifically, it looks for installed components (like Mono Framework, Java SDK, and so on) in order to verify which of them have to be downloaded and installed. Once this is done, another dialog appears on top of the window shown in Figure 1-4. Its header tells you "Thank you for downloading Visual Studio." In this dialog, you simply press the *Continue* button.

Figure 1-4. *Visual Studio installer is inspecting the operating system*

At this point, the Visual Studio installer might prompt you to install Xcode (Figure 1-5). This happens only if you do not have Xcode already installed. According to this dialog, you can install Xcode concurrently with the Visual Studio installation. Note that the Xcode installation is optional and depends on your current system configuration. I assume that you start with a clean install of macOS and therefore explicitly show how to install Xcode.

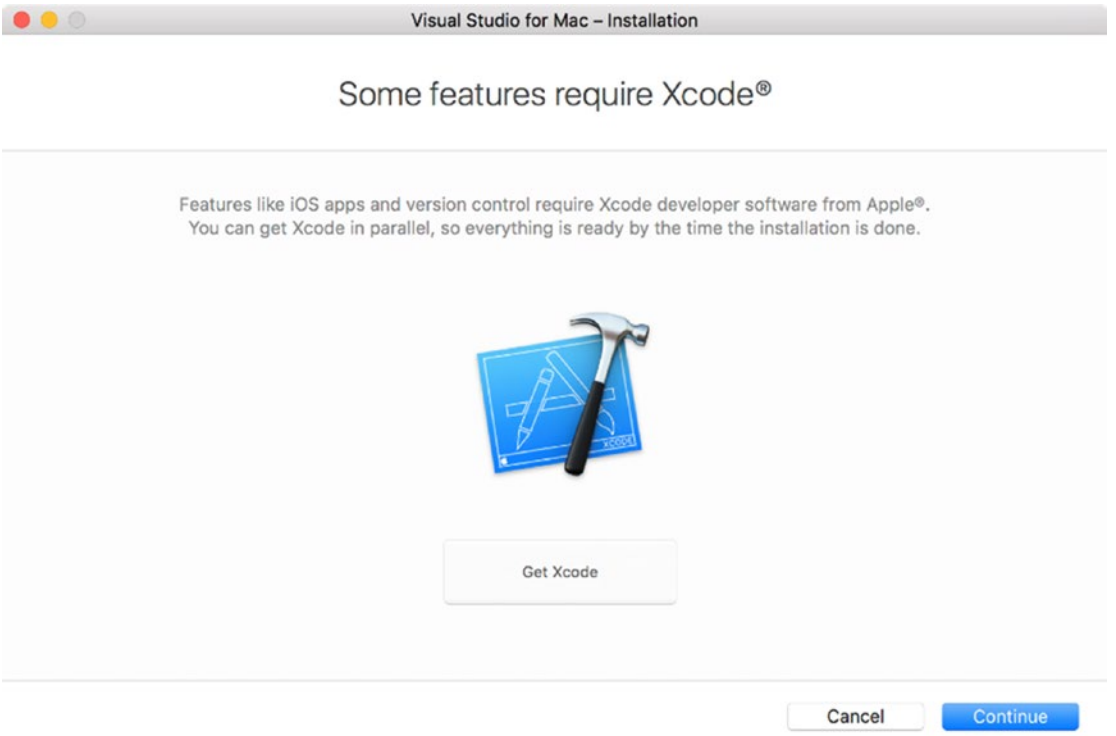


Figure 1-5. *A dialog prompting for Xcode installation*

To install Xcode, you can press the *Get Xcode* button shown in Figure 1-5. This will direct you to a website, where you click the *View in Mac App Store or Install App* button. It opens the Xcode page in the App Store, on which you only need to click the *Install App* button (Figure 1-6). Alternatively, to install Xcode you can open the Mac App Store locally and then look up Xcode. Irrespective of which method you choose, Xcode and all related developer tools will be downloaded and installed in the background. So, you can now go back to Visual Studio installer.

Figure 1-6. *Xcode page in the Mac App Store*

The Visual Studio installer will now let you choose which components to install (Figure 1-7). To reduce installation size, I uncheck the "Android + Xamarin.Forms" entry and only install iOS- and macOS-related components. Then, after you click the *Install* button, the actual installation process begins.

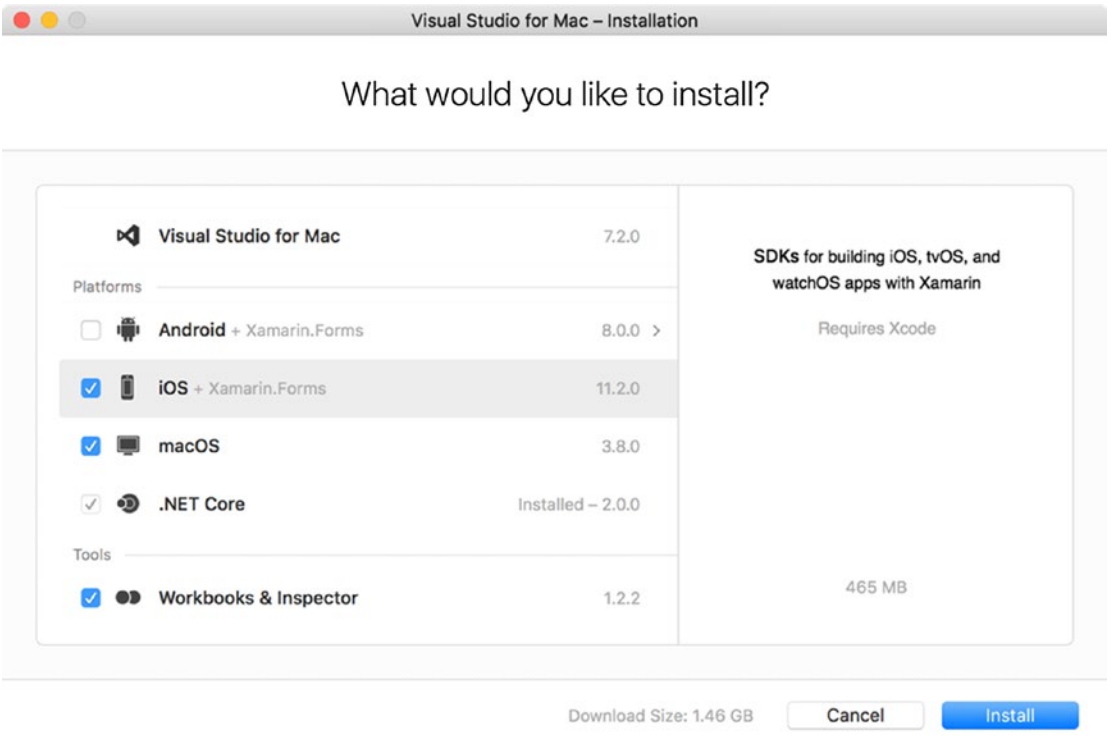


Figure 1-7. *Choosing components to install*

Visual Studio will now download and install the components. This will take a while, depending on network speed. You will be informed about each installation step and the overall progress, as shown in Figure 1-8. Also, as depicted in this figure, macOS may prompt you for the administrator password several times during installation. Once installation has finished, an appropriate dialog appears. Note that to build and run apps in the simulator, you will need to wait until Xcode installation has finished.

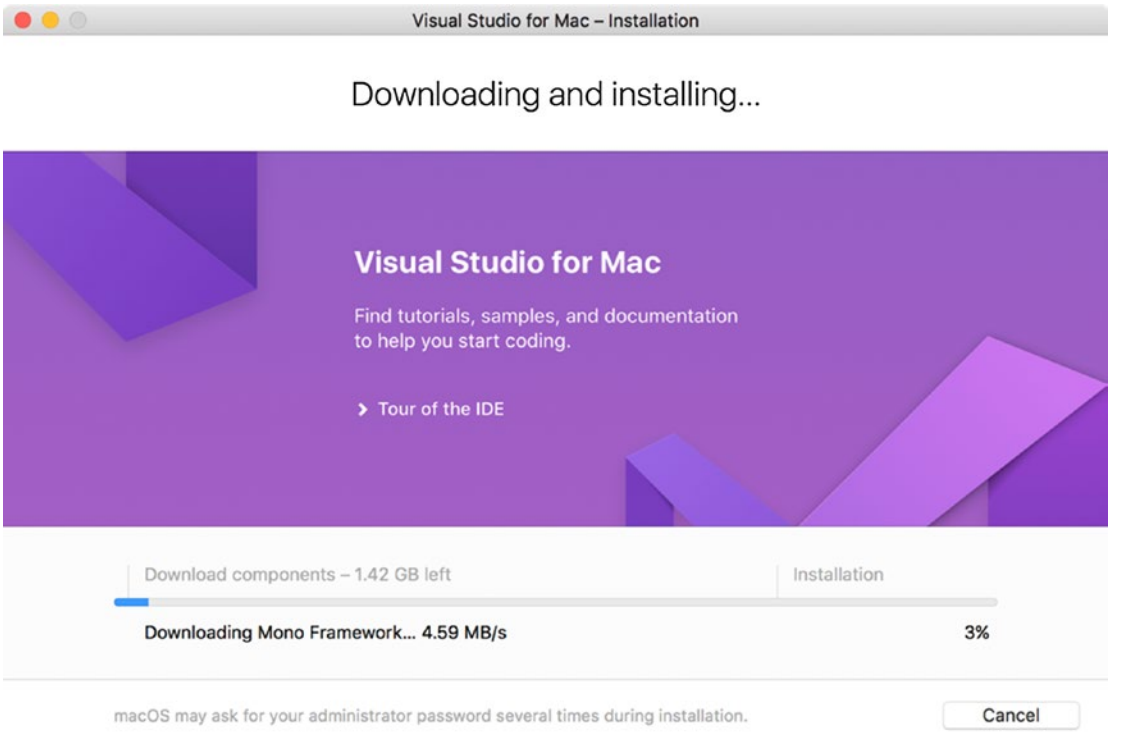


Figure 1-8. *Installing Visual Studio for Mac*

Hello, World! App

After installing the development tools, we can start building the first app. To jumpstart the Xamarin.iOS development, I will tell you how to create the project using the Single View app template. Then, we will supplement the app with a single button. This button will react to taps such that the native alert will be displayed. Subsequently, we will add specific actions to this alert. Displaying alerts is a typical functionality of not only introductory apps, but also real apps, where it is used to collect user input or get confirmation for performing irreversible operations.

Creating the Project

To create the project, open Visual Studio for Mac. A welcome screen, depicted in Figure 1-9, appears. Then, you either choose File/New Solution in the menu bar or click the *New Project* button, located under the "Recent" header. This activates the New Project window, shown in Figure 1-10.

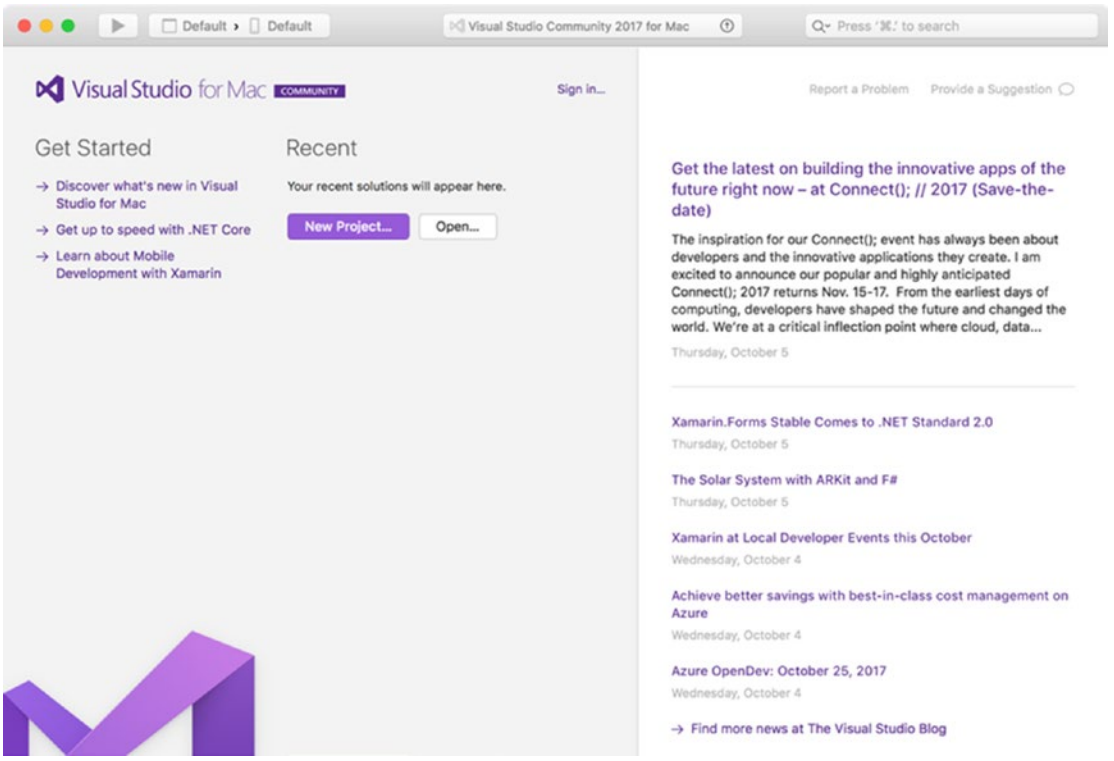


Figure 1-9. *A welcome screen for Visual Studio for Mac*

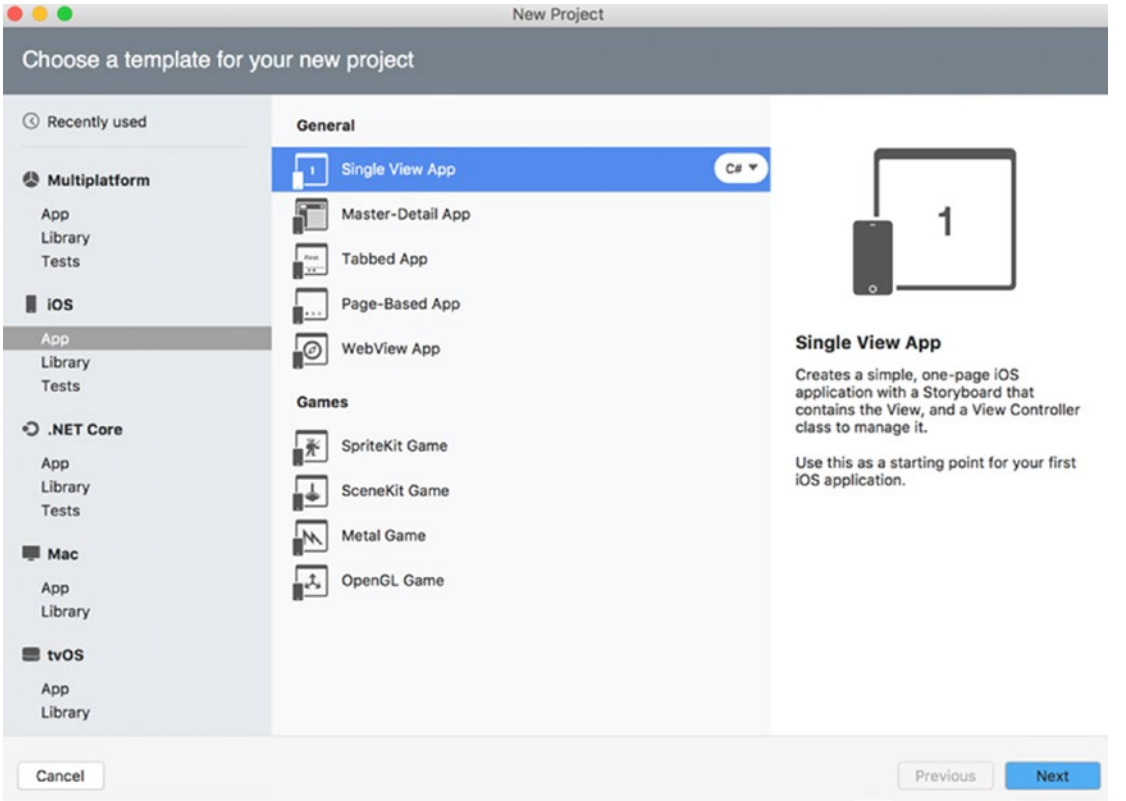


Figure 1-10. *Project template selection*

The New Project creator lets you choose a template for your project. To filter the list of templates to items directly related to iOS apps, you click *App entry* under the iOS tab. A list of available project templates will then appear on the right. This list is divided into two categories: General and Games. In this book, we will only use project templates from the General group. This category contains the following templates:

- Single View App — You use this template to create the app, which comprises a single view; i.e., an app without any navigation, like the app shown in Figure 1-1.
- Master-Detail App — This template creates apps that use Master-Detail interface. In such cases, a list displays short descriptions of objects. Once you choose an object from this list (master), corresponding details will be displayed in the dedicated area (detail). Master-Detail interface is used, for example, in the Stocks iOS app (Figure 1-11).