

4 STEPS TO JUMPSTART YOUR XAMARIN APP DEVELOPMENT

Mobile applications have quickly become vital tools for businesses and can play a pivotal role in their success. Consumer-facing apps can bolster audience engagement and the customer experience, while internal apps can significantly improve productivity and operations.












At the same time, the mobile market has never been more competitive. People have expectations for the mobile apps they use, so companies need high-quality apps to stand out from the crowd. For customer-facing apps, the user experience determines whether they stick around. [As a study conducted by Appboy found](#), more than 75% of users stop using an app after the second day. For internal apps, the user experience can impact usage and productivity. If an app is cumbersome, productivity could suffer and employees may even start looking for workarounds and rogue solutions.

Regardless of who companies are targeting, the message is clear: If the app doesn't deliver, users are quick to jump ship. It's no longer good enough to simply have an app. Businesses need well-designed and well-developed apps that function as intended across all mobile platforms and deliver a seamless user experience.

While there is no secret formula for building a successful app, nearly every developer will agree that choosing the appropriate development environment is a pivotal first step in the process. For many companies, Xamarin has become an attractive option. Because Xamarin uses a C# shared codebase, teams can use Xamarin to build cross-platform mobile apps with native performance, all while using the skills and expertise they've already developed.

This whitepaper will help you kickstart your Xamarin app development process by walking you through your initial steps. After reading this starting guide, you should have a better understanding of the essential tooling for Xamarin development and a clear path for building an elegant user interface for your cross-platform apps.

Mobile Development Landscape

- Go Mobile Web  
- Go Native   
- Go Web Hybrid  
- Go Cross-Compiled 
- Go JS Native   

Step 1: Choose Your IDE

When it comes to developing applications, developers are often required to work with a variety of tools and coding languages. Fortunately, Integrated Development Environments (IDEs) simplify the task. For Xamarin development, there are two rich IDEs to choose from: [Visual Studio for Windows](#) and [Xamarin Studio for Mac](#).

Both Visual Studio and Xamarin Studio come in completely free yet feature-rich community editions. They also sport some common features for ease of development, such as:

- Rich Intellisense support for code completion
- Intelligent typing assist for iOS and Android API mappings
- Built-in Xamarin project templates
- Easy code navigation and IDE configurations
- Choice of various device emulators with varying screen sizes

However, there are some key differences that you should be aware of, as these discrepancies may impact which IDE you go with. Some limitations to keep in mind are:

- Visual Studio on Windows will need a Mac to act as an iOS build host running XCode—a part of the Apple licensing requirement. The Mac could be hard wired to the Windows development machine, somewhere on the network or in the cloud. In fact, with the recent [iOS Windows Simulator](#), you do not even have to leave the comforts of Visual Studio—the simulator will stream the iOS emulator from the Mac.
- Xamarin Studio on Mac can only target iOS and Android, but not Windows. The app solution from Xamarin Studio could, however, be brought over to Visual Studio on a Windows development machine later to add a Windows-supporting project, while still leveraging the common Portable Class Library (PCL) for code sharing between platforms.

At the end of the day, you should go with the IDE that your development team is most comfortable working with. While Xamarin Studio is no longer required to build iOS apps, factors such as interface preferences could influence how well your team works. Opt for the IDE that enables you to be the most productive.

Step 2: Arm Your Arsenal with Essential Tooling

Once you have chosen which IDE you'll be using for your Xamarin development, it's time to arm the arsenal with essential tooling. While many of the following tools aren't necessary, they'll often make your life much easier and help you build apps more effectively. On top of that, most are usable within the comforts of whichever IDE you go with.

NuGet: As the de facto package manager for the .NET ecosystem, NuGet will faithfully serve your Xamarin projects. Bring in your favorite .NET libraries, services and utility wrappers.

Xamarin Component Store: Your Xamarin app needs to be sleek and elegant, but you don't have to reinvent the wheel to deliver a sophisticated app. Augment your app with polished UI components, service SDKs and various cloud/social integrations—all from one component store.

MVVM Light: Bring some structure and sanity to your Xamarin apps by following the established Model-View-ViewModel (MVVM) pattern and including the MVVM Light framework. You get out-of-the-box features like commanding, messenger, inversion of control (IoC) and INotifyPropertyChanged UI data binding implementations.

Prism for Xamarin.Forms: Have you used and liked Prism for past XAML development? Now you can bring Prism capabilities to Xamarin.Forms development with a NuGet package and an optional [Prism Template pack](#) to get you started. You get to leverage built-in MVVM features like commanding, ViewModelLocator, event aggregator, navigation and IoC with Unity.

XUnit: This is the perfect open-source unit testing framework for your Xamarin apps, both for PCLs and device-specific projects. Simply get a NuGet package in your project, write some unit tests and run them against your app, either through the command line, MSBuild or using the iOS/Android test runners.

[Xamarin WorkBooks](#): Need a playground so you can experiment with native API features and see your tentative code in action as you build your Xamarin app? You're in luck, thanks to the newly introduced Xamarin Workbooks—a standalone application that enables you to build interactive documents with executable live code outside of any app. You could run code-fenced C# blocks inline or inside iOS/Android simulators. It's perfect for trying out or sharing new features before adding them to your app.

[Xamarin Test Cloud](#): Your professional Xamarin apps should not be at the mercy of mobile device idiosyncrasies. And let's face it, there are a plethora of devices across various platforms that may run your Xamarin app. The solution is Xamarin Test Cloud. You can automate app testing on 2,000+ real devices, all running in the cloud. Increase confidence in your app by testing user interactions on real devices to find bugs, with complete memory and performance analysis. Get polished reports, fix problems and repeat—that's how you ship high-quality apps.

[Prebuilt Apps](#): Jumpstart your Xamarin app development with a polished, pre-built app and showcase apps with open-source code in GitHub. Peruse unique features or incorporate UI components to give your Xamarin apps a flying start.

Adding these tools to your library can help improve the application development process in a variety of ways, from improved testing to easier integration. Make sure you use them as appropriate to build cohesive, comprehensive Xamarin apps.

Step 3: Embrace Open Source

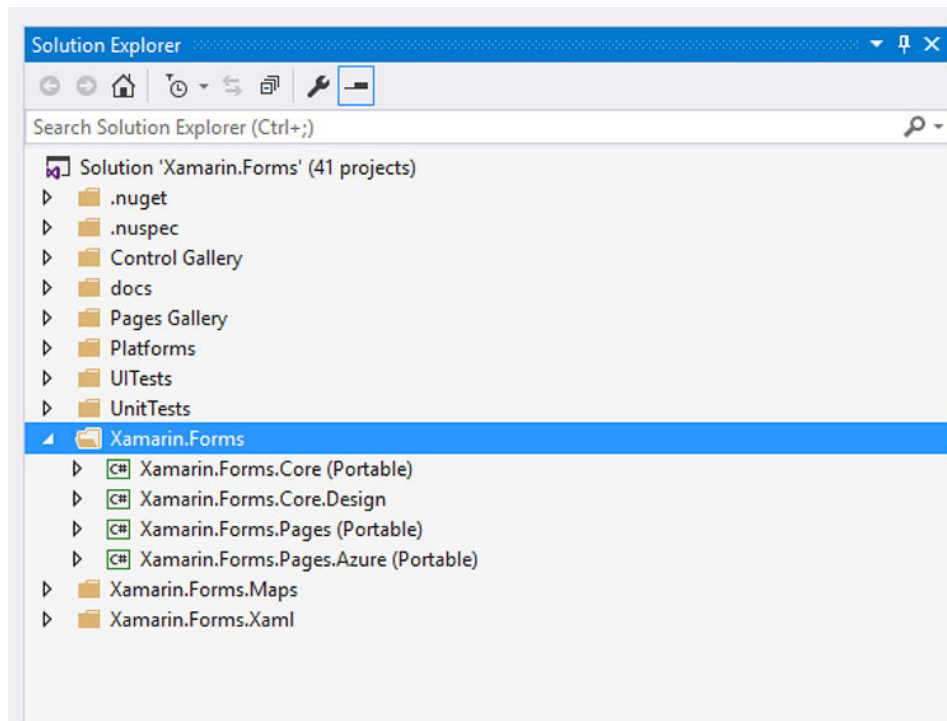
As if the Microsoft acquisition of Xamarin wasn't a big enough deal already, sweeter news greeted .NET developers in early 2016—Xamarin is now open source! All of the Xamarin frameworks—namely, Xamarin.iOS, Xamarin.Android and Xamarin.Forms—are completely open source, as are all the SDKs under a broad MIT license.

Xamarin going open source has two major repercussions. First, Xamarin and all of its tooling are now completely free. Second, there is now a true collaborative open-source community being built around Xamarin—and you can be a part of it.

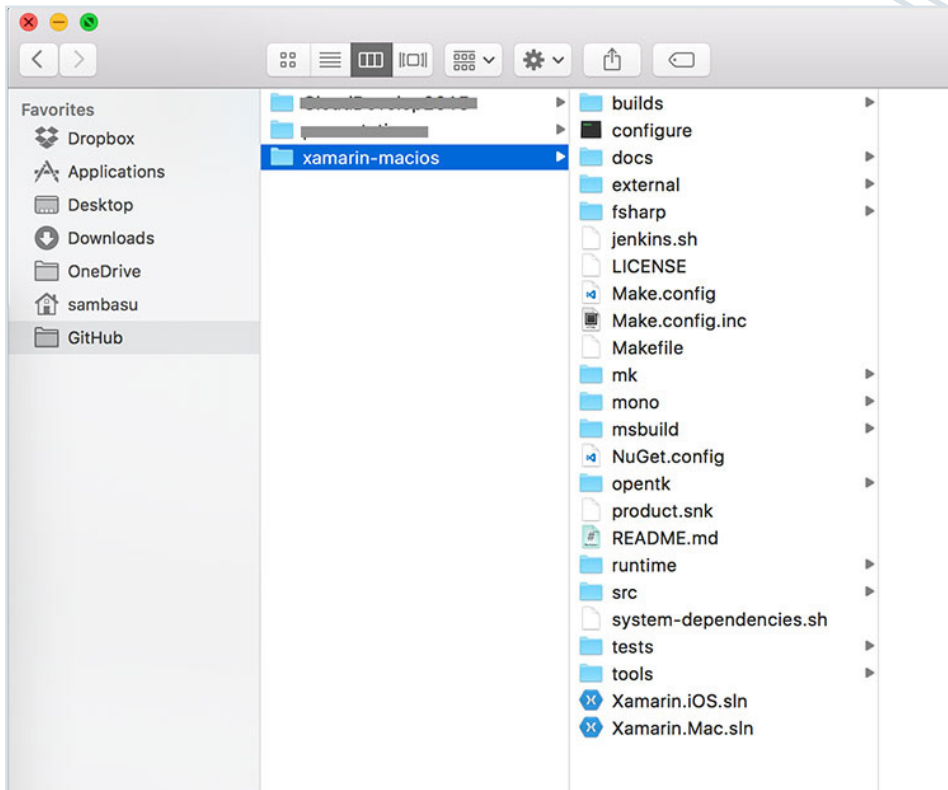
Why and How to Build from Source

Normally, you would just download the Xamarin SDKs and start building your app. But the true developer in you wants to take a peek under the covers and discover what this framework does for you. Now, you can look through the Xamarin source code, pull it down and build the SDK/libraries yourself.

Let's say you want to look deeper into Xamarin.Forms—you can [find the source on GitHub](#). If you want, you can download or clone the whole repository and build it locally. Follow the [defined requirements](#), open the solution in Visual Studio 2015 and fire up the build. Here's a glimpse of the Xamarin.Forms solution and component projects in VS:



Say you wanted to play around with Xamarin.iOS and Xamarin.Mac—you can start by [finding the source on GitHub once again](#). With compilation for Mono under the covers, Xamarin for iOS/Mac has a few more [defined requirements](#) in order to build locally. So get yourself the CLI tools and specific versions of XCode/Xamarin Studio and Mono. Here's what all the project components look like when pulled down:



There are helpful [scripts](#) that will install and provision dependencies. Once they are ready, fire up the build and proceed with the local installation:

```
$ make world  
$ make install-system
```

Give Back to the Open-Source Community

Now that you have been brave enough to look into the Xamarin source code and build the tooling/frameworks locally, take the next logical step forward and give back to the community. Think you can tweak something to be better, fix a bug or take a stab at a pending to-do item? This is your opportunity to pay back some technical debt and make the world a better place by contributing to open-source projects. Xamarin powers millions of developers, so there is a process you have to follow to contribute:

- To file a bug and track issues, [go through the official form](#)
- To take part in design discussions, subscribe to forms-devel@lists.xamarin.com or macios-devel@lists.xamarin.com
- To join the community, head over to [the Gitter chat room](#)
- To make appropriate code changes following the coding patterns, sign the [NET Foundation CLA](#) and make a pull request

When Xamarin cast off its commercial roots and went open source, it opened the door to new possibilities. Embrace the open-source community and take advantage of it to build better cross-platform apps.

Step 4: Build Your UI

You have started building your dream Xamarin app, picked the IDE of your choice and grabbed the necessary frameworks. Now, you're all set on the backend. However, that's only half the battle—your end users do not see any of that magic happening behind the scenes. What they do see is your app's UI, which directly impacts the user experience.

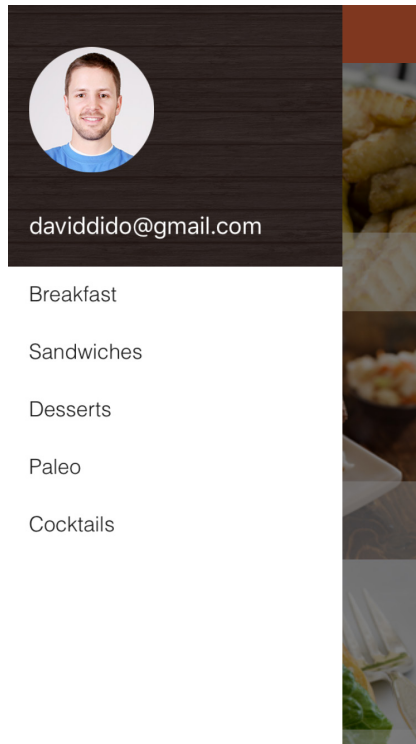
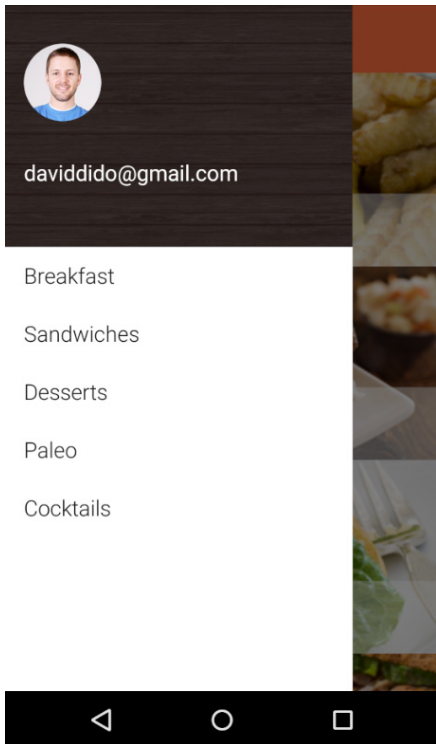
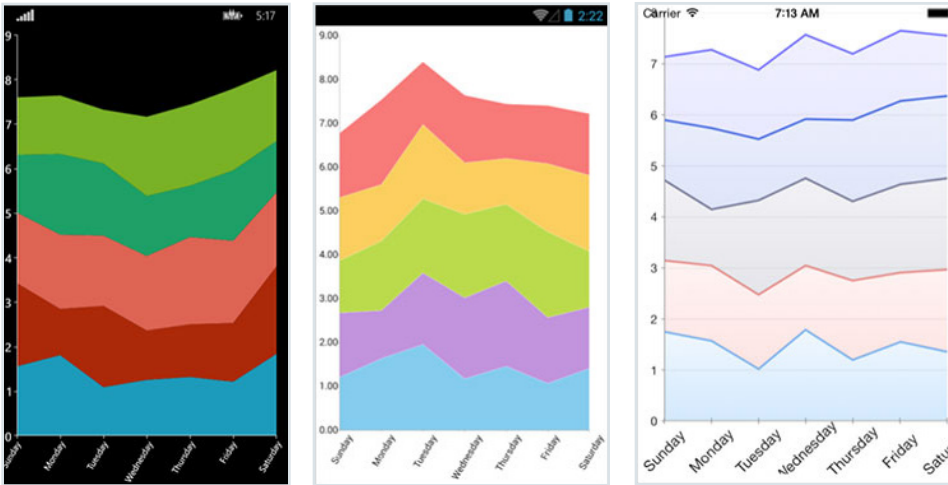
A quality UI can change the entire perception of your application. Whether your app is for employees or customers, an app that doesn't load properly or responds sluggishly can leave a bad first impression. Customers will just close the app and forget about it if the UI is unfriendly for users. Employees may even try to implement rogue solutions or workarounds just to avoid using the app. This is why you need to nail the UI—at the very least, a smooth, slick UI gets your foot in the door.

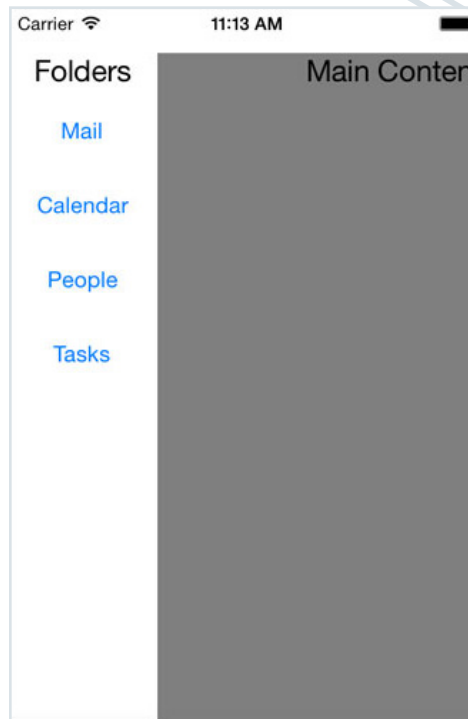
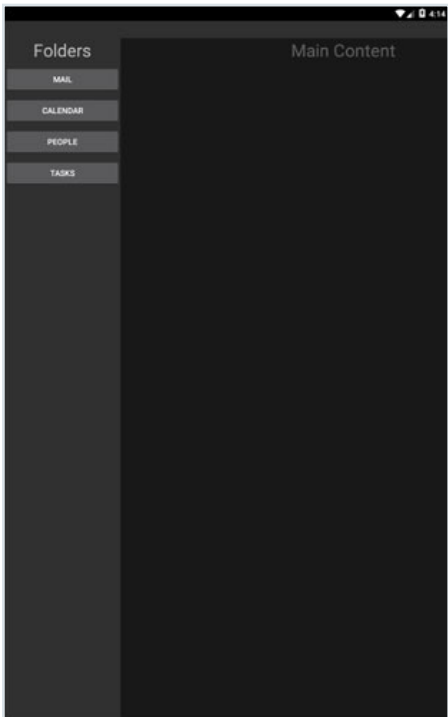
As you look to build your UI, you can start from scratch and aim to produce your own unique experience. But you don't have to do that—you could easily go for some well-engineered UI controls and get them straight out of the box.

Meet Telerik UI for Xamarin

Telerik® UI for Xamarin by Progress includes elegant, polished and performant native UI widgets for all your Xamarin apps. With ready-made UI components for Xamarin, you can deliver your Xamarin apps even quicker and delight users with beautiful, functional UI.

The widgets that come with UI for Xamarin are sophisticated and not easily made by hand. They include [charts](#), a [ListView](#), [SideDrawer](#) and much more.

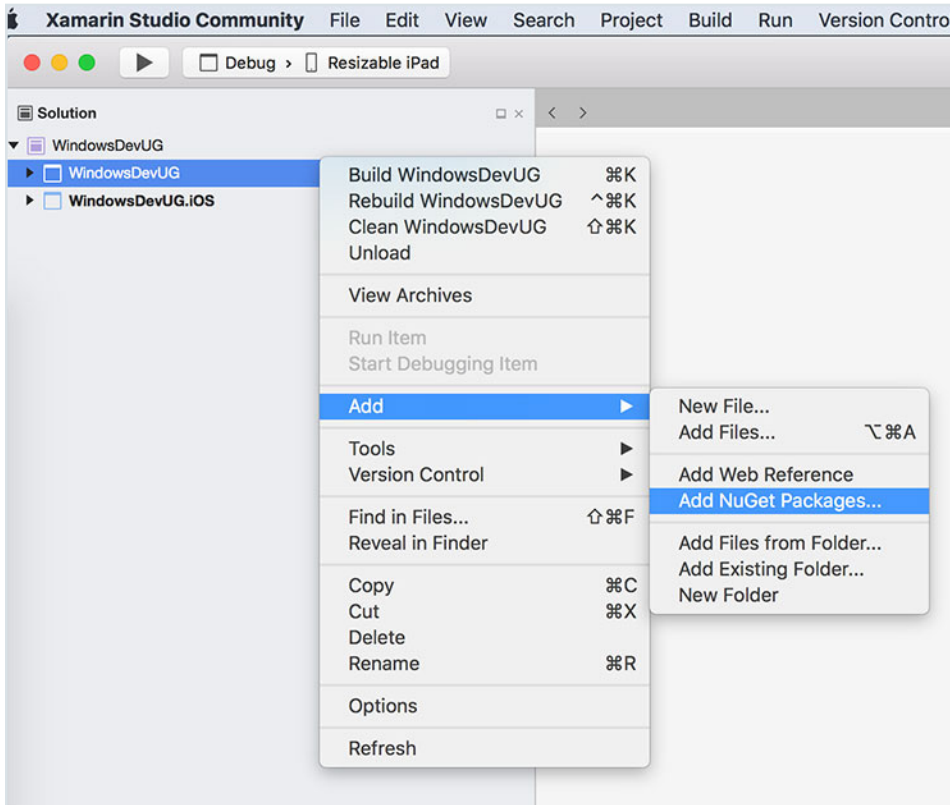
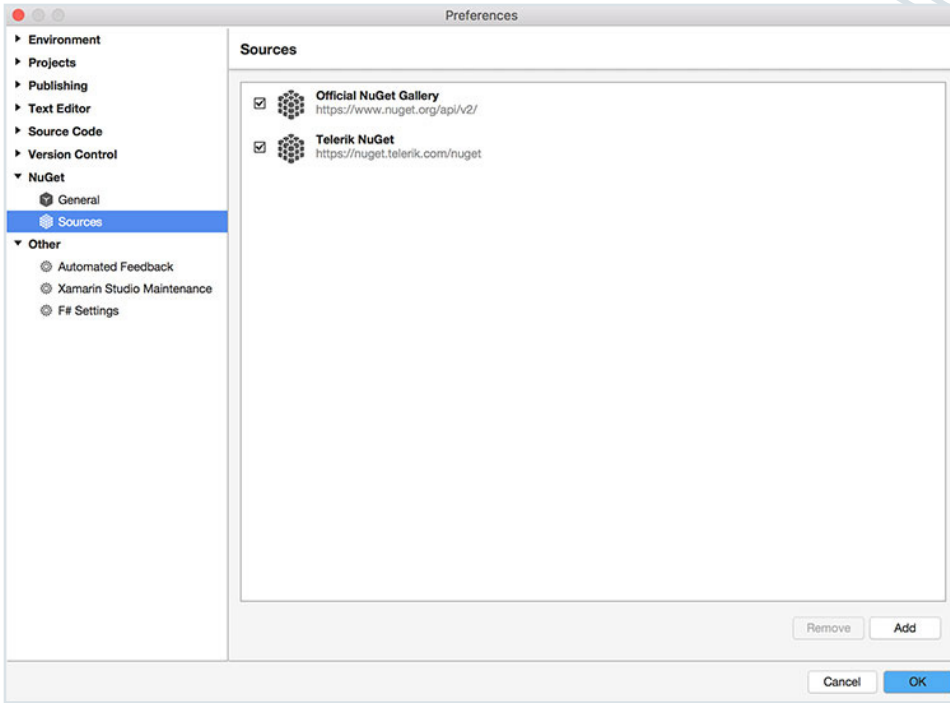


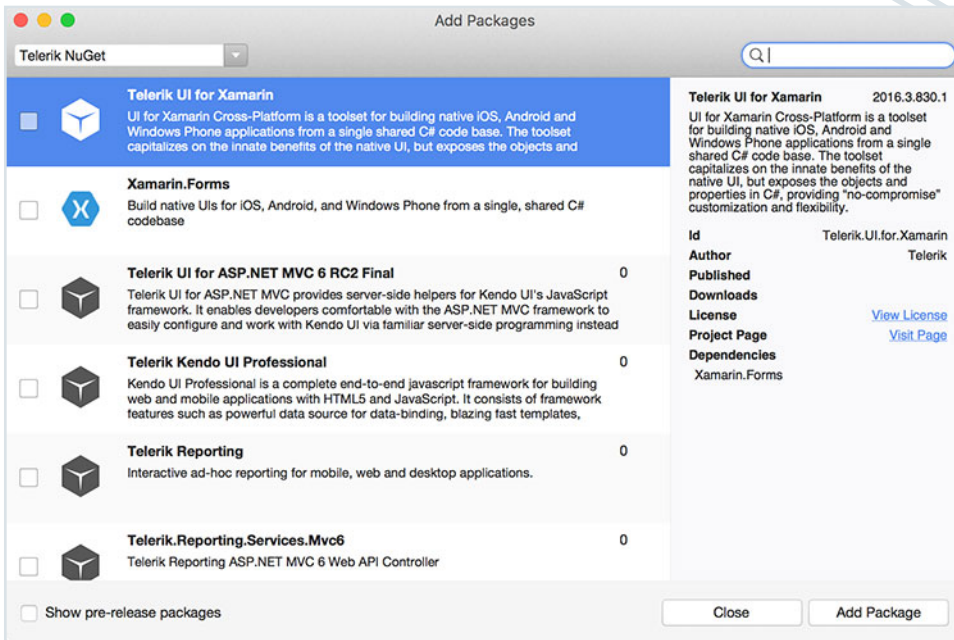


Incorporating UI for Xamarin into Your Project

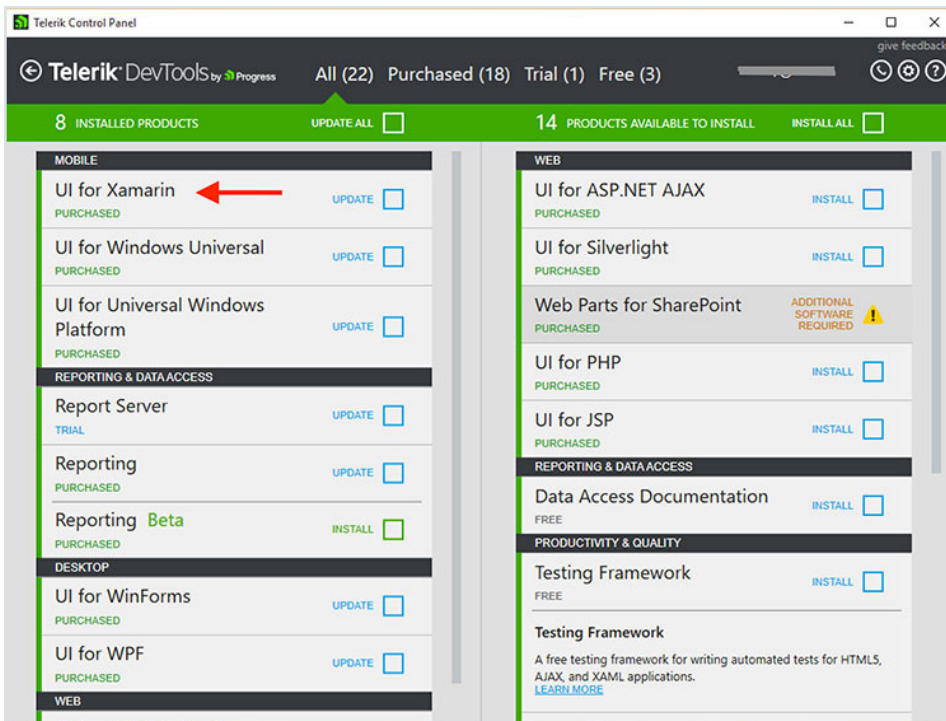
If you don't want to start from scratch, you should consider UI for Xamarin to help you get on your way. So, how do you incorporate UI for Xamarin into your Xamarin project? There are several ways you can go about doing that:

- 1. NuGet Package:** The NuGet route is probably the easiest way to include UI for Xamarin bits in your project. Simply point your NuGet source to the Telerik feed and you'll get a one-click install of UI for Xamarin. This works the same way from Visual Studio and Xamarin Studio, as shown below.

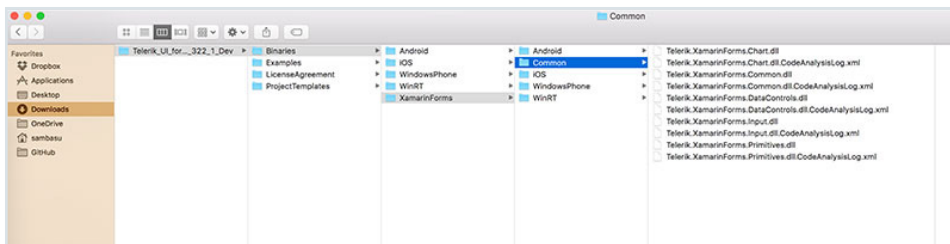
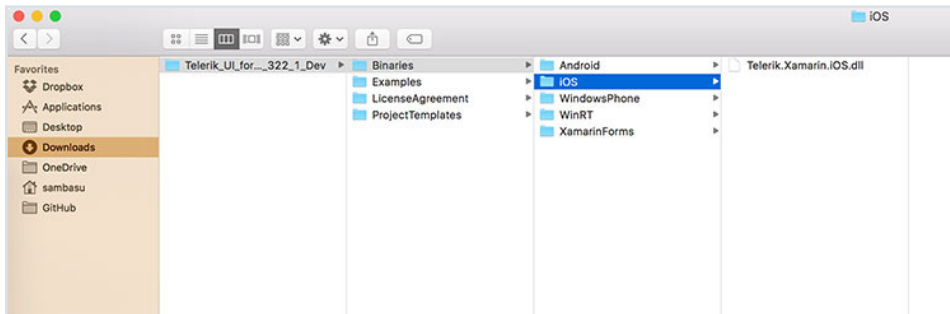




2. Telerik Control Panel: If you have an existing Telerik account subscription, you can leverage the Telerik Control Panel to download the UI for Xamarin bits. It will show up whether you just have a UI for Xamarin subscription or the entire Telerik DevCraft™ product suite, as seen below.



3. Plain Download: If you are just trying UI for Xamarin out, a simple trial download may be the easiest way to get started. Simply head to the [download site](#) and pull down the raw bits. Once downloaded, you'll find examples, templates and binaries specific to each platform. Make sure to check out the [documentation](#) for the UI widgets you are using—each has specific requirements to get the references right in your device-specific or shared projects.



Conclusion

A sleek and sophisticated mobile app is one of the best ways to ensure people use and find value in your app. But your apps—regardless of whether they are for customers or employees—need go above and beyond the benchmark to make a meaningful impact. This starts in the development process by making sure you choose the right development environment so you can build performant apps that work regardless of which platform your users are on.

Xamarin provides an excellent way for .NET developers to leverage their existing skills and build truly native cross-platform mobile apps. With an open-source foundation, mature tooling, supporting frameworks and polished UI libraries, the world is your oyster. You can use these tools to get up and running quickly, and develop an app that will leave a great first impression on users.




About Progress

Progress (NASDAQ: PRGS) is a global leader in application development, empowering the digital transformation organizations need to create and sustain engaging user experiences in today's evolving marketplace. With offerings spanning web, mobile and data for on-premise and cloud environments, Progress powers startups and industry titans worldwide, promoting success one customer at a time. Learn about Progress at www.progress.com or 1-781-280-4000.

Worldwide Headquarters

Progress, 14 Oak Park, Bedford, MA 01730 USA Tel: +1 781 280-4000 Fax: +1 781 280-4095

On the Web at: www.progress.com

Find us on  facebook.com/progresssw  twitter.com/progresssw  youtube.com/progresssw

For regional international office locations and contact information, please go to www.progress.com/worldwide

Progress, Telerik and DevCraft are trademarks or registered trademarks of Progress Software Corporation and/or one of its subsidiaries or affiliates in the U.S. and/or other countries. Any other trademarks contained herein are the property of their respective owners.

© 2016 Progress Software Corporation and/or its subsidiaries or affiliates.

All rights reserved.

Rev 2016/11 | 161013-0078

