**Introduction:**

It is time to learn xamarin cross platform development and people are already started in so many organizations, because of we can deliver native iOS, Android, and Windows apps using existing skills, teams, and code(C#, xaml ) with free of cost. Many company's are started hiring the Xamarin developers, and if you are good at xamarin, they are ready to give bigger packages for you as much as possible. If you are looking for Xamarin developer position in new company and you may need to face interview questions on Xamarin Technology.

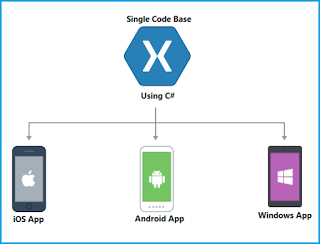
And you can also read nearly 50 Xamarin interview questions from my original [blog](http://bsubramanyamraju.blogspot.com/2016/07/top-50-xamarin-cross-platform.html)

**1. What is the difference between Native App and Cross Platform APP?**

* Native App:If we want to develop native apps, we need to choose the specific native language (C#, Objective-C, Swift, and Java) for developing specific native app for Windows, Android, iOS.
* Cross Platform App:This is the way to develop all three mobile apps using single code base wrapped in a native application layer, but it always requires customization of interfaces to native languages.

**2. What is Xamarin?**

It is cross platform development technology, where we can build native user interfaces for iOS, Android and Windows Phone using single codebase with C#.

[](https://4.bp.blogspot.com/-G1ZE8xZQlIo/V3OnAPKLJaI/AAAAAAAACsI/RN3RQ1RUXJcGi6VrGGVftoH-A9_rRmwBACLcB/s1600/3.%2BXamarinForms.PNG)

**3. Xamarin history?**

* **Xamarin** is a Microsoft-owned San Francisco, California based software company founded in May 2011 by the engineers that created Mono, Mono for Android and MonoTouch .
* **Novell had granted a perpetual license for Mono** on july 2011 for MONO
* **On May 16, 2011:** Miguel de Icaza announced on his blog that Mono would be developed and supported by Xamarin,
* **December 2012, Xamarin released Xamarin.Mac** which allows developers to build C#-based applications for the Apple OS X operating system and package them for publishing via the Apple App Store.
* **Xamarin 2.0 was released in February 2013 Xamarin.Android and Xamarin.iOS** that make it possible to do native Android, iOS and Windows development in C#, with either Visual Studio or Xamarin Studio.
* **Xamarin.froms:** Introduced in Xamarin 3 on May 28, 2014 and allows one to use portable controls subsets that are mapped to native controls of Android, iOS and Windows Phone.
* **On February 24, 2016:** Microsoft announced it signed a definitive agreement to acquire Xamarin
* **On March 30, 2016:** Microsoft acquired the xamarin. And made it for all with free of cost.

**4. Mono vs. Monodevelop vs Xamarin**

**Mono:** It is a framework and an open source implementation of Microsoft’s .NET Framework based on the open standards of European Computer Manufacturer's Association (**ECMA**) for the C# language and the Common Language Runtime. And Mono allows C# developers to write cross platform code targeting Windows, Mac OS X, Linux, Android, and iOS.

**MonoDevelop:** It is an open source integrated development environment (**IDE**) for Linux, OS X, and Windows. Its primary focus is development of projects that use Mono and .NET frameworks.

**Xamarin:** a subsidiary of Microsoft - is a company founded by the engineers who created Mono. Xamarin is the primary maintainer and commercial sponsor of Mono. It provides professional developer tools that make cross platform code easier to author, test, and maintain.

**5. What are the programming languages that support Xamarin Development?**

* 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#)
* Also offers xaml support for creating user interface.

**6. Why Xaml in Xamarin.Forms?**

* XAML(**Extensible Application Markup Language**) allows developers to define user interfaces in Xamarin.Forms applications using markup rather than code.
* XAML defines the visual appearance of a UI, and an associated code-behind file defines the logic. The UI design can be adjusted without necessarily making changes to the logic in code-behind. XAML in this role simplifies the workflow between individuals who might have a primary visual design responsibility and individuals who are responsible for app logic and information design.

**7. What are disadvantage of Xaml in Xamarin.Forms?**

* XAML cannot contain code. All event handlers must be defined in a code file.
* XAML cannot contain loops for repetitive processing. (However,ListView —can generate multiple children based on the objects in its ItemsSource collection.)
* XAML cannot contain conditional processing (However, a data-binding can reference a code-based binding converter that effectively allows some conditional processing.)
* XAML generally cannot instantiate classes that do not define a parameterless constructor. (However there is sometimes a way around this restriction.)
* XAML generally cannot call methods. (Again, this restriction can sometimes be overcome.)

**8. What are the IDE's we can use for Xamarin App Development?**

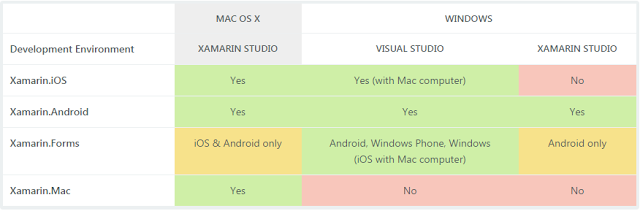
**Xamarin Studio:**This is Xamarin’s C# developer environment that runs on both Windows and Mac. Xamarin Studio has many of the features of Microsoft’s Visual Studio and uses exactly the same formats as Visual Studio: solutions (.sln) and C# projects (.csproj).

**Visual Studio:** Xamarin offers a Visual Studio extension/plugin for visual studio 2012, visual studio 2013 update 1. And now Xamarin development is directly available from Visual studio 2015.

**9. What was the major advantage of Xamarin Development?**

* We can deliver native iOS, Android, and Windows apps using existing skills, teams, and code(C#, xaml )
* We can take full advantage of Native API’s with Xamarin Technology . And Xamarin.Forms elements map to native controls and behaviors
* Xamarin Component Store allows adding high-quality components to your app directly from your IDE, including controls, web service APIs and more.
* PCL/ Shared projects makes very easy for developers to share the same code base across different projects.

**10. What are difference between Xamarin Studio and Visual Studio?**

[](https://3.bp.blogspot.com/-G2FpdpFBRbs/V3PC-LuxhJI/AAAAAAAACsY/fz4rY3bnPuATiI5QHKCx39tk1ZpKYdFkQCLcB/s1600/VSvsXamarinStudio.PNG)

**Notes:**

* We can't create Windows Phone, Windows & UWP from Xamarin Studio.
* Xamarin Studio in Mac can allows you to develop iOS & Android apps.
* Xmarin Studio in Windows can allows you to develop Android apps only. And it encourage Windows users to move to Visual Studio.
* Xamarin for Visual Studio supports any VS 2015, VS 2013 Update 2 and Visual Studio 2012 with non-Express editions. And visual studio can't install on Windows machine.
* To test iOS app on Windows OS with VS, must be a Mac computer accessible on same network.
* To develop Xamarin.Forms apps for the Universal Windows Platform (UWP) requires Windows 10 with Visual Studio 2015

**11. What are the life cycle's of Xamarin.forms app development?**

Life-Cycle means a specific sequence of the application from start to finish of the application. So there are three important states in the life cycle of a Xamarin app development.

**OnStart:**Called when the application starts.

**OnSleep:** Called each time the application goes to the background.

**OnResume:** Called when the application is resumed, after being sent to the background.

**Note:**

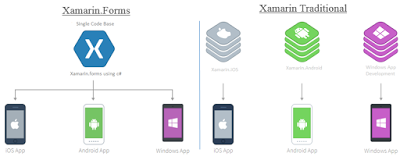
There is no method for application **termination**. Under normal circumstances (ie. not a crash) application termination will happen fom the OnSleep state, without any additional notifications to your code.

**12.How many ways we can create Xamarin applications?**

There are two ways to deliver native iOS, Android, and Windows apps using Xamarin Technology

**Traditional Xamarin approach:**This approach canprovide direct access to platform-specific API’s. And we can create platform specific apps such as using Xamarin.iOS for iOS applications, using Xamarin.Android we can create Android applications.

**Xamarin.forms:**This architecture is the same as that of traditional cross-platform apps. The most used method is to implement Portable Libraries or Shared Projects to save the shared code, and then create specific applications for each platform that will consume this shared code.

[](https://3.bp.blogspot.com/-O5LNqrgfyRs/V3PTfUKGp6I/AAAAAAAACsw/DXTbsxddatEwYtKwAHQuqTWqJEdOLJs7gCLcB/s1600/XamarinTypes1.PNG)

**13. When to use Xamarin.Forms vs Xamarin Native?**

**Xamarin.Forms:**

* Apps that require little platform-specific functionality.
* Apps where code sharing is more important than custom UI.

**Xamarin.IOS,Xamarin.Android:**

* Apps that require specialized interactions.
* Apps with highly polished design.
* Apps that use many platform-specific APIs.
* Apps where custom UI is more important than code sharing.

**14. What are different extension output files generated from xamarin.forms?**

If we run the Xamarin app, it can generate following application packages for different platforms

**.xap/.Appx** for windows phone 8, Windows, UWP

**.ipa** file for IOS

**.apk** file for Android

**15. What are the different application package running modes in Xamarin?**

When you compile an application, you usually choose between below two modes:

**Debug:** Debug mode pads the executable file with extra information used for debugging purpose. After your application is bug-free and feature-complete, there's no longer any need to keep the debug information.

**Release:** Compiling in Release mode gives you a more compact (and even slightly faster) executable that doesn't include unnecessary internal debugging symbols

**Note:** Packages should be in release mode, which would be provided to an App store.

**16. How does Xamarin.Android(Mono for Android) work?**

On Android, Xamarin ships a fully functional implementation of the .NET runtime, called Mono, bundled with your app so that your code executes with all of the power of C# and .NET, including JIT-compilation, memory management, reflection, and the .NET base class libraries. Developers can use any native features by interacting with .NET class libraries that provide one-to-one mappings to all of the native APIs on Android.

**17. What is the difference between Xaml & axml in Xamarin Technology?**

* AXML and XAML are two different XML specifications.
* AXML is just supported/available for Xamarin.Android.
* XAML is the way Xamarin Forms could standardize Cross Platform UI based on XML specification.
* XAML is the way Xamarin Forms could standardize Cross Platform UI based on XML specification.
* If you are using native Xamarin Android you will do UI using axml, if using Forms then using XAML.

**18. How does Xamarin.IOS work?**

On iOS, Xamarin uses Mono, a fully functional implementation of the .NET runtime, to fully compile your app into a native ARM executable ahead of time (AOT) so that your code executes with all of the power of C# and .NET, including memory management, reflection, and the .NET base class libraries. Developers can use any native features by interacting with .NET class libraries that provide one-to-one mappings to all of the native APIs on iOS, while complying with Apple's security restrictions which prevent execution of dynamically generated code on device.

**19. What is the project structure of Xamarin.Forms?**

If we create App with xamarin.forms project using xamarin studio or visual studio. Then created project will have bellow structure:

* **Shared:** Shared Project containing the code common to all projects and it will be either PCL or Shared Project.
* **AppAndroid:** Xamarin.Android application project.
* **AppiOS:** Xamarin.iOS application project.
* **AppWinPhone:** Windows Phone application project.
* **AppWindows:** Windows application project.
* **AppUWP:** Universal Windows Platform application project.

**20. What are the different code sharing techniques in xamarin.forms?**

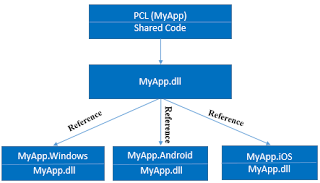
There are two alternative methods for sharing code between cross-platform applications:

* Portable Libraries(PCL)
* Shared Projects

The goal of a code-sharing strategy is to support the architecture, where a single codebase can be utilized by multiple platforms.

**21. Difference between Portable Class Library (PCL) & Shared Projects?**

**Portable Class Library (PCL):**

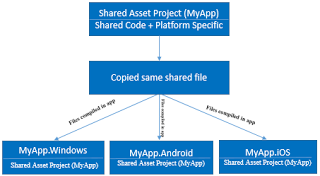
**[](https://2.bp.blogspot.com/-L0FnL8N9yos/V3UeZbEwEeI/AAAAAAAACtw/QzUhSHfTkWw4nB43DlROnPb63z_qm56GwCKgB/s1600/PCL.PNG)**

- Produce libraries that can be shared across multiple platform.

- Targeting specific platform includes interfaces

- Refactoring operations always update.

**Shared Projects:**

**[](https://3.bp.blogspot.com/-DDks3qQCIO8/V3UepJ01yJI/AAAAAAAACt4/HAb25Jk31cQ0eRt7NTNMT4AFdjJNFJb3wCLcB/s1600/SAP.PNG)**

- Shared code can be branched based on the platform using compiler directives( eg. using #if \_\_ANDROID\_\_, #if \_\_IOS\_\_, #if WINDOWS\_PHONE\_APP, #if WINDOWS\_APP..etc)

- No output assembly.

- Refactoring operations ,not update automatically.

**22. What does Pages in Xamarin.Forms?**

The Page class is a visual element that occupies most or all of the screen and contains a single child. And pages in xamarin.forms are directly mapped to like below:

* In iOS  it is a View Controller
* In Windows Phone it is a Page.
* In Android it is a Activity.

**23. What are the different kinds of Pages in Xamarin.Forms?**

Fortunately Xamarin.Forms support multiple page representations, And every representation is having its own purpose like below:

* **ContentPage:**A Page that displays a single View, often a container such as a StackLayout or a ScrollView.
* **MasterDetailPage:**A Page that manages two panes of information.
* **NavigationPage:**A Page that manages the navigation and user-experience of a stack of other pages.
* **TabbedPage:**A Page that allows navigation between children pages, using tabs.
* **TemplatedPage:**A Page that displays full-screen content with a control template, and the base class for ContentPage.
* **CarouselPage:**A Page allowing swipe gestures between subpages, like a gallery.

**24. What are the layout controls available in Xamarin.Forms?**

Layout is the process of sizing and positioning child elements in UI.

* **ContentPresenter:** A layout manager for templated views. Used within a ControlTemplate to mark where the content to be presented appears.
* **ContentView:** An element with a single content. ContentView has very little use of its own. Its purpose is to serve as a base class for user-defined compound views.
* **Frame:** An element containing a single child, with some framing options. Frame have a default Xamarin.Forms.Layout.Padding of 20.
* **ScrollView:**An element capable of scrolling if it's Content requires.
* **TemplatedView:** An element that displays content with a control template, and the base class for ContentView.
* **AbsoluteLayout:** Positions child elements at absolute requested positions. User assigned anchors and bounds defines the position and size of the control.
* **Grid:** A layout containing views arranged in rows and columns.
* **RelativeLayout:** A Layout that uses Constraints to layout its children.
* **StackLayout:** A Layout that positions child elements in a single line which can be oriented vertically or horizontally. This layout will set the child bounds automatically during a layout cycle. User assigned bounds will be overwritten and thus should not be set on a child element by the user.

**25. What is Views?**

Views are commonly known as controls of widgets. And they can refer to visual objects such as Label, Button, Entry, Image, BoxView, ListView, TableView..etc. All UI elements are typically are sub-classes of View.

**26. What is is the difference between ListView & TableView?**

* The ListView and TableView controls are so similar, you can think of them as a single control. The major difference between the two is the manner in which they lay out their items, and it’s easy to change the layout so each control emulates the other.
* The **ListView**control displays its data stacked vertically, much like a standard listbox. Use this control to display an ordered list of data, especially long lists that require scrolling like a list of email messages, a list of contacts, or search results.
* The **TableView**control displays its data stacked horizontally in rows (although you can alter this behavior and have it displayed in columns first, as well). You use this control when you need more space for rich visualization of each item to be displayed.
* One of the big differences is ListView provides you a ItemsSource and a Itemtemplate and TableView does not. So items must be added as children manually.

**27. How to draw rectangle in Xamarin.Forms?**

BoxView is used to draw a solid colored rectangle. So below source code can create Accent colored rectanlge with width ->200 & Height->100

1. <BoxView Color="Accent"  WidthRequest ="200"  HeightRequest = "100"/>

**28. What is the difference between Editor & Entry?**

Both views are helpful for user to enter text, but there is a small difference is **Entry**only has one line, whereas a **Editor**usually has multiple lines that allows user to press ENTER.

**29. What is Cell in Xamarin.Forms?**

Simple it is not a visual element, but it just describes a template for creating a visual element. And one important note is that Cells are elements designed to be added to **ListView**or **TableView**controls.

**30. What is difference between Entry and EntryCell in Xamarin.Forms?**

* **Entry** is a visual element, where user can enter single line text.
* **EntryCell**is not visual element, it is subtype of Cell and it describes a template for label and a single line text entry field. It is normally for use in a ListView or TableView.