

Xamarin .Forms Notes for Professionals

Chapter 3: Xamarin Forms Layouts Section 3.1: AbsoluteLayout

AbsoluteLayout positions and sizes child elements proportional to its own size and position. Child views may be positioned and sized using proportional values or static values, and proportional values can be mixed.



A definition of an AbsoluteLayout in XAML looks like this:

```
<AbsoluteLayout>
<label text="I'm centered on iPhone 4 but no other device"
AbsoluteLayout.LayoutBounds="113, 156, 100, 100"
AbsoluteLayout.LayoutFlags="PositionProportional" />
<button text="I'm better center on every device."
AbsoluteLayout.LayoutBounds="3,3, 5, 1"
AbsoluteLayout.LayoutFlags="PositionProportional" />
</AbsoluteLayout>
```

The same layout would look like this in code:

```
title = "Absolute Layout Example - Code";
var layout = new AbsoluteLayout();
var label = new Label {
    Text = "I'm centered on iPhone 4 but no other device.",
    LayoutFlags = LayoutFlags.PositionProportional,
    AbsoluteLayout.LayoutBounds = new Rectangle(113, 156, 100, 100);
};
layout.Children.Add(label);
```

Chapter 10: Display Alert

Section 10.1: DisplayAlert

An alert box can be popped up on a Xamarin Forms Page by the method, DisplayAlert. We can provide a Title, Body (text to be alerted) and one/two Action Buttons. Page offers two overloads of DisplayAlert method.

```
1. public Task DisplayAlert (String title, String message, String cancel)
```

This override presents an alert dialog to the application user with a single cancel button. The alert displays modally and once dismissed the user continues interacting with the application.

Example:

```
DisplayAlert ("Alert", "You have been alerted.", "OK");
```

Above snippet will present a native implementation of Alerts in each platform (AlertDialog in Android, UIAlertView in iOS, MessageBox in Windows) as below.



```
2. public System.Threading.Tasks.Task<bool> DisplayAlert (string title, string message, string accept, string cancel)
```

This override presents an alert dialog to the application user with an accept and a cancel button. It captures response by presenting two buttons and returning a boolean. To get a response from an alert, supply text for buttons and await the method. After the user selects one of the options the answer will be returned to the caller.

Example:

```
var answer = await DisplayAlert ("Question?", "Would you like to play a game?", "Yes", "No");
Debug.WriteLine ("Answer: " + (answer ? "Yes" : "No"));
```

Example 2: If Condition true or false check to alert proceed

Chapter 16: Data Binding Section 16.1: Basic Binding to ViewModel

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="MyAssembly.EntryPage"
xmlns:viewModel="MyAssembly.ViewModel"
xmlns:layout="MyAssembly.Layout"
xmlns:binding="MyAssembly.Binding"
xmlns:stack="MyAssembly.Stack"
xmlns:vertical="MyAssembly.Vertical"
xmlns:horizontal="MyAssembly.Horizontal"
xmlns:options="MyAssembly.Options"
xmlns:spacing="MyAssembly.Spacing"
xmlns:label="MyAssembly.Label"
xmlns:entry="MyAssembly.Entry"
xmlns:button="MyAssembly.Button"
xmlns:command="MyAssembly.Command"
xmlns:binding2="MyAssembly.Binding2"
xmlns:myviewModel="MyAssembly.MyViewModel"
xmlns:myviewmodel="MyAssembly.MyViewModel"
xmlns:myviewmodel2="MyAssembly.MyViewModel2"
xmlns:myviewmodel3="MyAssembly.MyViewModel3"
xmlns:myviewmodel4="MyAssembly.MyViewModel4"
xmlns:myviewmodel5="MyAssembly.MyViewModel5"
xmlns:myviewmodel6="MyAssembly.MyViewModel6"
xmlns:myviewmodel7="MyAssembly.MyViewModel7"
xmlns:myviewmodel8="MyAssembly.MyViewModel8"
xmlns:myviewmodel9="MyAssembly.MyViewModel9"
xmlns:myviewmodel10="MyAssembly.MyViewModel10"
xmlns:myviewmodel11="MyAssembly.MyViewModel11"
xmlns:myviewmodel12="MyAssembly.MyViewModel12"
xmlns:myviewmodel13="MyAssembly.MyViewModel13"
xmlns:myviewmodel14="MyAssembly.MyViewModel14"
xmlns:myviewmodel15="MyAssembly.MyViewModel15"
xmlns:myviewmodel16="MyAssembly.MyViewModel16"
xmlns:myviewmodel17="MyAssembly.MyViewModel17"
xmlns:myviewmodel18="MyAssembly.MyViewModel18"
xmlns:myviewmodel19="MyAssembly.MyViewModel19"
xmlns:myviewmodel20="MyAssembly.MyViewModel20"
xmlns:myviewmodel21="MyAssembly.MyViewModel21"
xmlns:myviewmodel22="MyAssembly.MyViewModel22"
xmlns:myviewmodel23="MyAssembly.MyViewModel23"
xmlns:myviewmodel24="MyAssembly.MyViewModel24"
xmlns:myviewmodel25="MyAssembly.MyViewModel25"
xmlns:myviewmodel26="MyAssembly.MyViewModel26"
xmlns:myviewmodel27="MyAssembly.MyViewModel27"
xmlns:myviewmodel28="MyAssembly.MyViewModel28"
xmlns:myviewmodel29="MyAssembly.MyViewModel29"
xmlns:myviewmodel30="MyAssembly.MyViewModel30"
xmlns:myviewmodel31="MyAssembly.MyViewModel31"
xmlns:myviewmodel32="MyAssembly.MyViewModel32"
xmlns:myviewmodel33="MyAssembly.MyViewModel33"
xmlns:myviewmodel34="MyAssembly.MyViewModel34"
xmlns:myviewmodel35="MyAssembly.MyViewModel35"
xmlns:myviewmodel36="MyAssembly.MyViewModel36"
xmlns:myviewmodel37="MyAssembly.MyViewModel37"
xmlns:myviewmodel38="MyAssembly.MyViewModel38"
xmlns:myviewmodel39="MyAssembly.MyViewModel39"
xmlns:myviewmodel40="MyAssembly.MyViewModel40"
xmlns:myviewmodel41="MyAssembly.MyViewModel41"
xmlns:myviewmodel42="MyAssembly.MyViewModel42"
xmlns:myviewmodel43="MyAssembly.MyViewModel43"
xmlns:myviewmodel44="MyAssembly.MyViewModel44"
xmlns:myviewmodel45="MyAssembly.MyViewModel45"
xmlns:myviewmodel46="MyAssembly.MyViewModel46"
xmlns:myviewmodel47="MyAssembly.MyViewModel47"
xmlns:myviewmodel48="MyAssembly.MyViewModel48"
xmlns:myviewmodel49="MyAssembly.MyViewModel49"
xmlns:myviewmodel50="MyAssembly.MyViewModel50"
xmlns:myviewmodel51="MyAssembly.MyViewModel51"
xmlns:myviewmodel52="MyAssembly.MyViewModel52"
xmlns:myviewmodel53="MyAssembly.MyViewModel53"
xmlns:myviewmodel54="MyAssembly.MyViewModel54"
xmlns:myviewmodel55="MyAssembly.MyViewModel55"
xmlns:myviewmodel56="MyAssembly.MyViewModel56"
xmlns:myviewmodel57="MyAssembly.MyViewModel57"
xmlns:myviewmodel58="MyAssembly.MyViewModel58"
xmlns:myviewmodel59="MyAssembly.MyViewModel59"
xmlns:myviewmodel60="MyAssembly.MyViewModel60"
xmlns:myviewmodel61="MyAssembly.MyViewModel61"
xmlns:myviewmodel62="MyAssembly.MyViewModel62"
xmlns:myviewmodel63="MyAssembly.MyViewModel63"
xmlns:myviewmodel64="MyAssembly.MyViewModel64"
xmlns:myviewmodel65="MyAssembly.MyViewModel65"
xmlns:myviewmodel66="MyAssembly.MyViewModel66"
xmlns:myviewmodel67="MyAssembly.MyViewModel67"
xmlns:myviewmodel68="MyAssembly.MyViewModel68"
xmlns:myviewmodel69="MyAssembly.MyViewModel69"
xmlns:myviewmodel70="MyAssembly.MyViewModel70"
xmlns:myviewmodel71="MyAssembly.MyViewModel71"
xmlns:myviewmodel72="MyAssembly.MyViewModel72"
xmlns:myviewmodel73="MyAssembly.MyViewModel73"
xmlns:myviewmodel74="MyAssembly.MyViewModel74"
xmlns:myviewmodel75="MyAssembly.MyViewModel75"
xmlns:myviewmodel76="MyAssembly.MyViewModel76"
xmlns:myviewmodel77="MyAssembly.MyViewModel77"
xmlns:myviewmodel78="MyAssembly.MyViewModel78"
xmlns:myviewmodel79="MyAssembly.MyViewModel79"
xmlns:myviewmodel80="MyAssembly.MyViewModel80"
xmlns:myviewmodel81="MyAssembly.MyViewModel81"
xmlns:myviewmodel82="MyAssembly.MyViewModel82"
xmlns:myviewmodel83="MyAssembly.MyViewModel83"
xmlns:myviewmodel84="MyAssembly.MyViewModel84"
xmlns:myviewmodel85="MyAssembly.MyViewModel85"
xmlns:myviewmodel86="MyAssembly.MyViewModel86"
xmlns:myviewmodel87="MyAssembly.MyViewModel87"
xmlns:myviewmodel88="MyAssembly.MyViewModel88"
xmlns:myviewmodel89="MyAssembly.MyViewModel89"
xmlns:myviewmodel90="MyAssembly.MyViewModel90"
xmlns:myviewmodel91="MyAssembly.MyViewModel91"
xmlns:myviewmodel92="MyAssembly.MyViewModel92"
xmlns:myviewmodel93="MyAssembly.MyViewModel93"
xmlns:myviewmodel94="MyAssembly.MyViewModel94"
xmlns:myviewmodel95="MyAssembly.MyViewModel95"
xmlns:myviewmodel96="MyAssembly.MyViewModel96"
xmlns:myviewmodel97="MyAssembly.MyViewModel97"
xmlns:myviewmodel98="MyAssembly.MyViewModel98"
xmlns:myviewmodel99="MyAssembly.MyViewModel99"
xmlns:myviewmodel100="MyAssembly.MyViewModel100"
/>
```

```
MyViewModel.cs:
using System;
using System.ComponentModel;
namespace MyAssembly.ViewModel
{
    public class MyViewModel : INotifyPropertyChanged
    {
        private string _name = String.Empty;
        private string _phone = String.Empty;
        public string Name
        {
            get { return _name; }
            set
            {
                if (_name != value)
                {
                    _name = value;
                    OnPropertyChanged(nameof(Name));
                }
            }
        }
        public string Phone
        {
            get { return _phone; }
            set
            {
                if (_phone != value)
                {
                    _phone = value;
                    OnPropertyChanged(nameof(Phone));
                }
            }
        }
    }
}
```

100+ pages
of professional hints and tricks

Contents

About	1
Chapter 1: Getting started with Xamarin.Forms	2
Section 1.1: Installation (Visual Studio)	2
Section 1.2: Hello World Xamarin Forms: Visual Studio	4
Chapter 2: Why Xamarin Forms and When to use Xamarin Forms	7
Section 2.1: Why Xamarin Forms and When to use Xamarin Forms	7
Chapter 3: Xamarin Forms Layouts	8
Section 3.1: AbsoluteLayout	8
Section 3.2: Grid	10
Section 3.3: ContentPresenter	11
Section 3.4: ContentView	12
Section 3.5: ScrollView	13
Section 3.6: Frame	14
Section 3.7: TemplatedView	14
Section 3.8: RelativeLayout	15
Section 3.9: StackLayout	16
Chapter 4: Xamarin Relative Layout	19
Section 4.1: Box after box	19
Section 4.2: Page with an simple label on the middle	21
Chapter 5: Navigation in Xamarin.Forms	23
Section 5.1: NavigationPage flow with XAML	23
Section 5.2: NavigationPage flow	24
Section 5.3: Master Detail Navigation	25
Section 5.4: Using INavigation from view model	26
Section 5.5: Master Detail Root Page	28
Section 5.6: Hierarchical navigation with XAML	29
Section 5.7: Modal navigation with XAML	31
Chapter 6: Xamarin.Forms Page	32
Section 6.1: TabbedPage	32
Section 6.2: ContentPage	33
Section 6.3: MasterDetailPage	34
Chapter 7: Xamarin.Forms Cells	36
Section 7.1: EntryCell	36
Section 7.2: SwitchCell	36
Section 7.3: TextCell	37
Section 7.4: ImageCell	38
Section 7.5: ViewCell	39
Chapter 8: Xamarin.Forms Views	41
Section 8.1: Button	41
Section 8.2: DatePicker	42
Section 8.3: Entry	43
Section 8.4: Editor	43
Section 8.5: Image	44
Section 8.6: Label	45
Chapter 9: Using ListView	47
Section 9.1: Pull to Refresh in XAML and Code behind	47

Chapter 10: Display Alert	48
Section 10.1: DisplayAlert	48
Section 10.2: Alert Example with only one button and action	49
Chapter 11: Accessing native features with DependencyService	50
Section 11.1: Implementing text-to-speech	50
Section 11.2: Getting Application and Device OS Version Numbers - Android & iOS - PCL	53
Chapter 12: DependencyService	55
Section 12.1: Android implementation	55
Section 12.2: Interface	56
Section 12.3: iOS implementation	56
Section 12.4: Shared code	57
Chapter 13: Custom Renderers	58
Section 13.1: Accessing renderer from a native project	58
Section 13.2: Rounded label with a custom renderer for Frame (PCL & iOS parts)	58
Section 13.3: Custom renderer for ListView	59
Section 13.4: Custom Renderer for BoxView	61
Section 13.5: Rounded BoxView with selectable background color	65
Chapter 14: Caching	68
Section 14.1: Caching using Akavache	68
Chapter 15: Gestures	70
Section 15.1: Make an Image tappable by adding a TapGestureRecognizer	70
Section 15.2: Gesture Event	70
Section 15.3: Zoom an Image with the Pinch gesture	78
Section 15.4: Show all of the zoomed Image content with the PanGestureRecognizer	78
Section 15.5: Tap Gesture	79
Section 15.6: Place a pin where the user touched the screen with MR.Gestures	79
Chapter 16: Data Binding	81
Section 16.1: Basic Binding to ViewModel	81
Chapter 17: Working with Maps	83
Section 17.1: Adding a map in Xamarin.Forms (Xamarin Studio)	83
Chapter 18: Custom Fonts in Styles	92
Section 18.1: Accessing custom Fonts in Syles	92
Chapter 19: Push Notifications	94
Section 19.1: Push notifications for Android with Azure	94
Section 19.2: Push notifications for iOS with Azure	96
Section 19.3: iOS Example	99
Chapter 20: Effects	101
Section 20.1: Adding platform specific Effect for an Entry control	101
Chapter 21: Triggers & Behaviours	105
Section 21.1: Xamarin Forms Trigger Example	105
Section 21.2: Multi Triggers	106
Chapter 22: AppSettings Reader in Xamarin.Forms	107
Section 22.1: Reading app.config file in a Xamarin.Forms Xaml project	107
Chapter 23: Creating custom controls	108
Section 23.1: Label with bindable collection of Spans	108
Section 23.2: Implementing a CheckBox Control	110
Section 23.3: Create an Xamarin Forms custom input control (no native required)	116
Section 23.4: Creating a custom Entry control with a MaxLength property	118

Section 23.5: Creating custom Button	119
Chapter 24: Working with local databases	121
Section 24.1: Using SQLite.NET in a Shared Project	121
Section 24.2: Working with local databases using xamarin.forms in visual studio 2015	123
Chapter 25: CarouselView - Pre-release version	133
Section 25.1: Import CarouselView	133
Section 25.2: Import CarouselView into a XAML Page	133
Chapter 26: Exception handling	135
Section 26.1: One way to report about exceptions on iOS	135
Chapter 27: SQL Database and API in Xamarin Forms.	137
Section 27.1: Create API using SQL database and implement in Xamarin forms.	137
Chapter 28: Contact Picker - Xamarin Forms (Android and iOS)	138
Section 28.1: contact_picker.cs	138
Section 28.2: MyPage.cs	138
Section 28.3: ChooseContactPicker.cs	139
Section 28.4: ChooseContactActivity.cs	139
Section 28.5: MainActivity.cs	140
Section 28.6: ChooseContactRenderer.cs	141
Chapter 29: Xamarin Plugin	144
Section 29.1: Media Plugin	144
Section 29.2: Share Plugin	146
Section 29.3: ExternalMaps	147
Section 29.4: Geolocator Plugin	148
Section 29.5: Messaging Plugin	150
Section 29.6: Permissions Plugin	151
Chapter 30: OAuth2	155
Section 30.1: Authentication by using Plugin	155
Chapter 31: MessagingCenter	157
Section 31.1: Simple example	157
Section 31.2: Passing arguments	157
Section 31.3: Unsubscribing	158
Chapter 32: Generic Xamarin.Forms app lifecycle? Platform-dependant!	159
Section 32.1: Xamarin.Forms lifecycle is not the actual app lifecycle but a cross-platform representation of it	159
Chapter 33: Platform-specific behaviour	161
Section 33.1: Removing icon in navigation header in Anroid	161
Section 33.2: Make label's font size smaller in iOS	161
Chapter 34: Platform specific visual adjustments	163
Section 34.1: Idiom adjustments	163
Section 34.2: Platform adjustments	163
Section 34.3: Using styles	164
Section 34.4: Using custom views	164
Chapter 35: Dependency Services	165
Section 35.1: Access Camera and Gallery	165
Chapter 36: Unit Testing	166
Section 36.1: Testing the view models	166
Chapter 37: BDD Unit Testing in Xamarin.Forms	172
Section 37.1: Simple Specflow to test commands and navigation with NUnit Test Runner	172

Section 37.2: Advanced Usage for MVVM	174
Credits	175
You may also like	176

[Click here to download full PDF material](#)