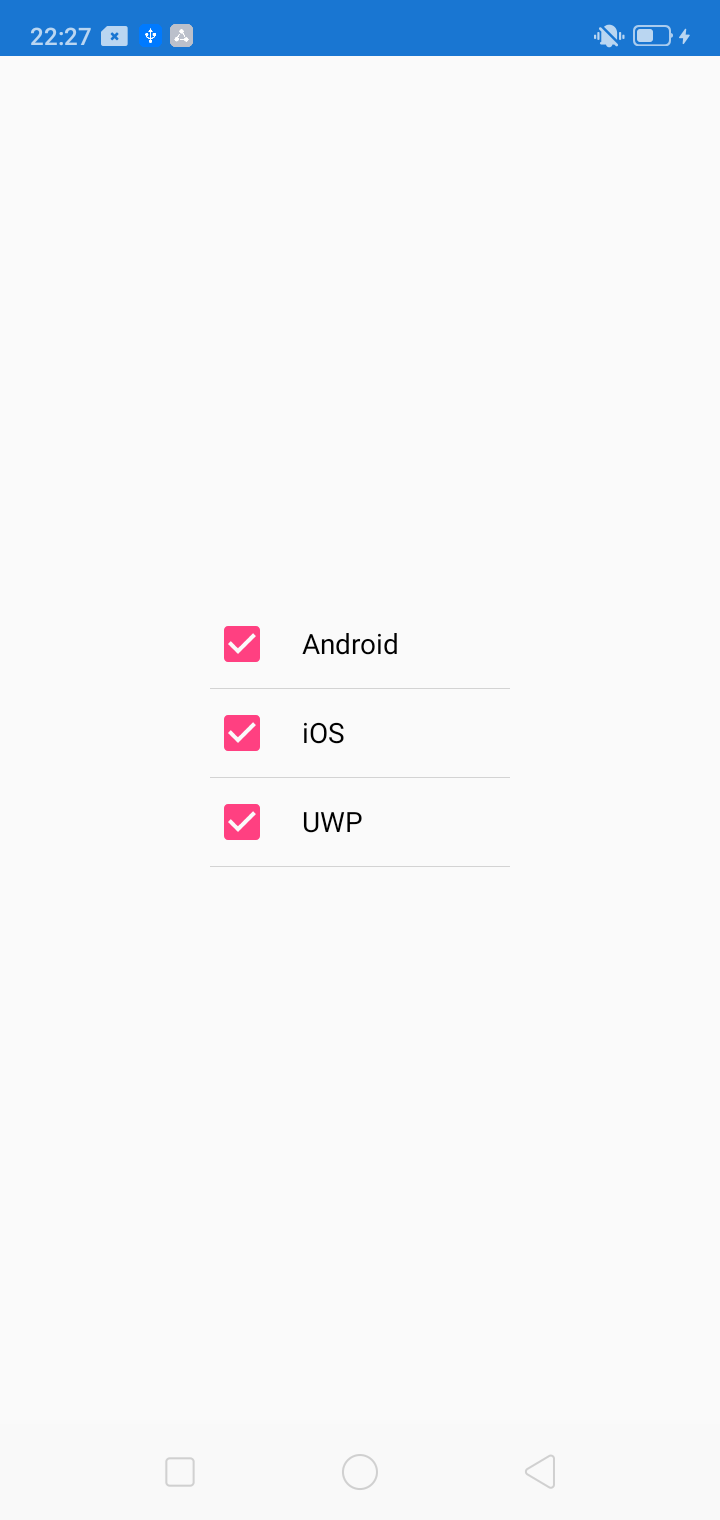
In this article, you will learn how to use a Bindable Layout in Xamarin.Forms.



Xamarin.Forms is an open-source UI framework that runs on multiple platforms with a single shared codebase. It allows developers to create user interfaces in XAML with code-behind in C#. These interfaces are rendered as performant native controls on each platform.

# Bindable Layout

If you are someone who is looking for a lightweight approach to display a small collection of items, however, do not wish to use [ListView](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.listview) or [CollectionView](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.collectionview) considering the memory and performance issues, then you are at the right place. You are looking for a Bindable Layout.

Bindable layouts enable any layout class to generate its content by binding to a collection of items. It provides an option to set and customize the appearance of each item with a [DataTemplate](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.datatemplate). Bindable layouts are provided by the BindableLayout class, which exposes the following attached properties:

* ItemsSource – specifies the list of items to be displayed.
* ItemTemplate – specifies the [DataTemplate](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.datatemplate) to apply to each item in the collection of items displayed.
* ItemTemplateSelector – specifies the [DataTemplateSelector](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.datatemplateselector) that will be used to choose a [DataTemplate](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.datatemplate) for an item at runtime.

In simple terms, a bindable layout is a small version of ListView to display a series of items with the same pattern. However, the only difference is that a Bindable Layout does not allow your items to scroll, unlike ListView.

### Prerequisites

* Visual Studio 2017 or later (Windows or Mac)

## Setting up a Xamarin.Forms Project

Let’s start by creating a new Xamarin.Forms project by following the below steps.

Visual Studio 2019 has more options in the launch view.

* Clone or check out the code from any repository
* Open a project or solution
* Open a local folder from your computer
* Create a new project.

Choose "Create a new project".

Visual Studio 2019 
Open recent 
Search recent (Alt+S) 
SiSystems.ClientApp.sln 
onesi.sln 
E:NDevNCAS 
FCMPushpoc.sln 
Sisystems.sln 
E:NDevNOneAp p 
Gateway.sln 
E:NDevNMatchGuidelWotnetpooI 
MatchGuide 
E:NDev 
MatchGuideWebAPl.sln 
11/21/2019 12:35 PM 
11/21/2019 12:03 PM 
11/18/2019 12:34 PM 
11/13/2019 5:17 PM 
11/5/2019 4:59 PM 
11/5/2019 4:58 PM 
11/5/2019 4:58 PM 
Get started 
Clone or check out code 
Get code from an online repository like GitHub 
or Azure DevOps 
Open a project or solution 
Open a local Visual Studio project or .sln file 
Open a local folder 
Navigate and edit code within any folder 
Create a new project 
Choose a project template with code scaffolding 
to get started 
Continue without code 

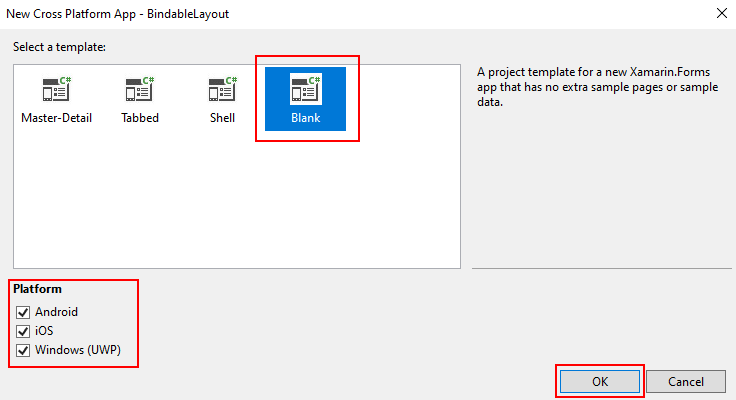
Now, filter by Project Type as Mobile and choose the Mobile App (Xamarin.Forms).

Create a new project 
Recent project templates 
A list of your recently accessed templates will be 
displayed here. 
Search for templates (Alt+S) 
All Languages 
All Platforms 
Mobile App (Xamarin.Forms) 
3 
A multiproject template for building apps for iOS and A 
All Project Types 
Extensions 
Games 
roid 
Library 
Machine 
Xamarin.Forms. 
C# Android iOS 
Android App (Xamarin) 
Windows 
Mobile 
2 
Project templates for creating Android phone and tablet apps ' 
C# Android Mobile 
iOS App (Xamarin) 
Plugin 
Roslyn 
Service 
Project templates for creating iOS apps for iPhone and iPad wi Test 
ios Mobile 
Android Wear App (Xamarin) 
A project for creating an Android Wear app with Xamarin. 
C# Android Mobile 
watchOS App (Xamarin) 
A project for creating a watchOS app with Xamarin. 
ios Mobile 
'"SIX Project 

Enter the project name of your wish. Usually, the project and solution name are the same for an app. Choose your preferred location for the project and click "Create".

Configure your new project 
Mobile App (Xamarin.Forms) At-droid jos Windows 
Project name 
Location 
Solution name 
App I 
Place solution and project in the same directory 
Mobile 

Select the Blank App and target platforms - Android, iOS and Windows (UWP).



Wait for the solution to load. Expand the solution using the Solution Explorer. By default, you can see 4 projects (.NET Standard, Android, iOS and UWP).

Expand the .NET Standard project and select the XAML page and double-click to open the MainPage.xaml page. You now have a basic Xamarin.Forms app. Press F5 or click the run button to try it out.

## Create a Bindable Layout

In this article, we will see how to create and use a Bindable Layout to display the list of platforms supported by Xamarin. For that first lets us create the model and view model classes required for binding to the view.

Create a new class called PlatformInfo.cs and declare the below properties.

|  |
| --- |
| public class PlatformInfo : INotifyPropertyChanged  {  private bool \_isChecked;  private string \_platformName;  public bool IsChecked  {  get { return \_isChecked; }  set { \_isChecked = value; NotifyPropertyChanged(); }  }  public string PlatformName  {  get { return \_platformName; }  set { \_platformName = value; NotifyPropertyChanged(); }  }  public event PropertyChangedEventHandler PropertyChanged;  public void NotifyPropertyChanged([System.Runtime.CompilerServices.CallerMemberName] string propertyName = "")  {  if (this.PropertyChanged != null)  this.PropertyChanged(this, new PropertyChangedEventArgs(propertyName));  }  } |

Create a new class called ViewModel.cs and write the below code.

|  |
| --- |
| public class ViewModel  {  public ViewModel()  {  this.GetContactsList();  }  public List<PlatformInfo> PlatformsList { get; set; }  private void GetContactsList()  {  if (this.PlatformsList == null)  this.PlatformsList = new List<PlatformInfo>();  this.PlatformsList.Add(new PlatformInfo() { IsChecked = true, PlatformName = "Android" });  this.PlatformsList.Add(new PlatformInfo() { IsChecked = true, PlatformName = "iOS" });  this.PlatformsList.Add(new PlatformInfo() { IsChecked = false, PlatformName = "UWP" });  }  } |

We have created the required collection and model object for binding to a Bindable Layout. Now, let's design a UI with a Bindable Layout to display the created list.

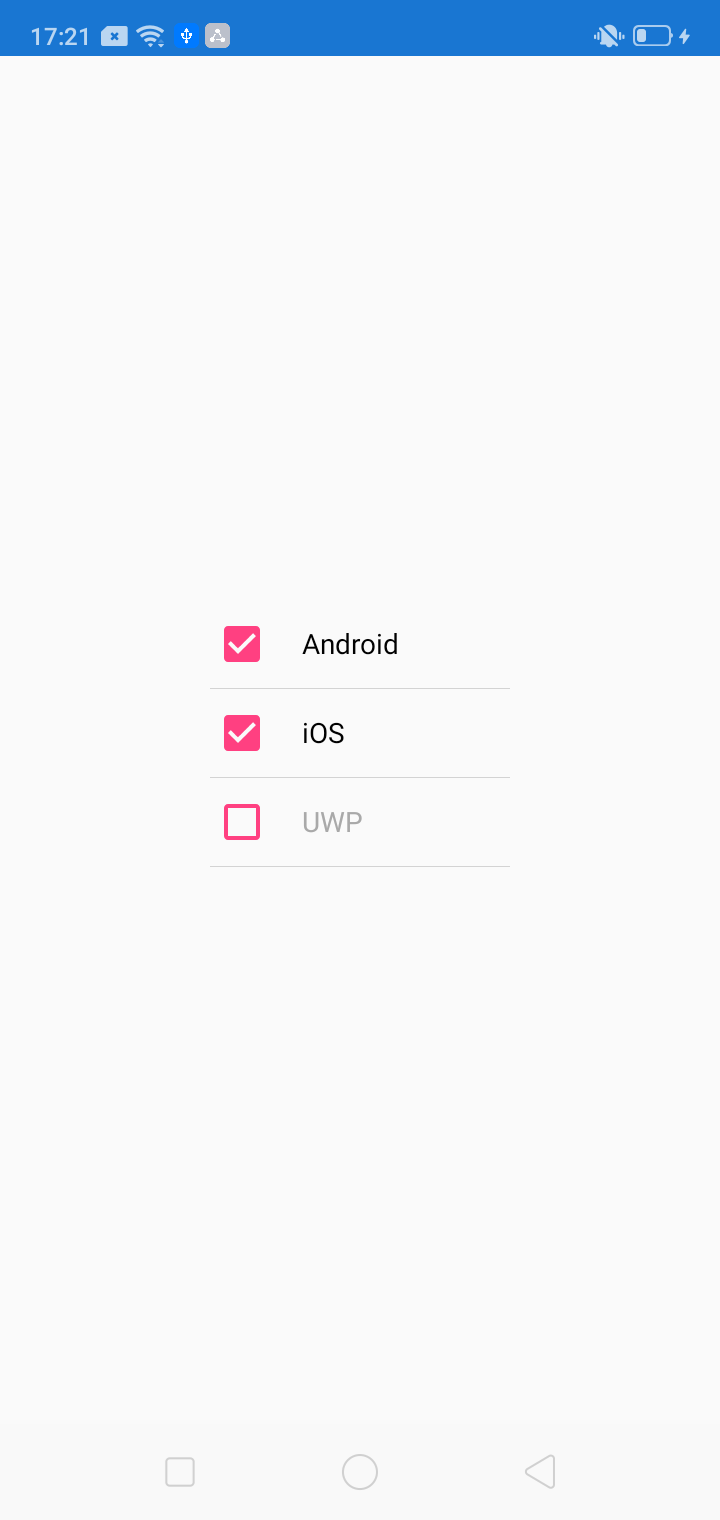
## Setting up the User Interface

Go to MainPage.Xaml and write the following code

MainPage.xaml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <ContentPage xmlns="http://xamarin.com/schemas/2014/forms"  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"  xmlns:d="http://xamarin.com/schemas/2014/forms/design"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:BindableLayout"  mc:Ignorable="d"  x:Class="BindableLayout.MainPage">  <ContentPage.BindingContext>  <local:ViewModel />  </ContentPage.BindingContext>  <StackLayout x:Name="contactList" BindableLayout.ItemsSource="{Binding PlatformsList}"  VerticalOptions="Center" HorizontalOptions="Center" WidthRequest="150">  <BindableLayout.ItemTemplate>  <DataTemplate>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="Auto"/>  <RowDefinition Height="0.5"/>  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="30" />  <ColumnDefinition Width="\*" />  </Grid.ColumnDefinitions>  <CheckBox IsChecked="{Binding IsChecked}" VerticalOptions="Center" />  <Label Grid.Column="1" TextColor="Black" Margin="10,0" Text="{Binding PlatformName}" IsEnabled="{Binding IsChecked}" VerticalOptions="Center">  <Label.Triggers>  <DataTrigger TargetType="Label" Binding="{Binding IsChecked}" Value="true">  <Setter Property="TextColor" Value="Black"/>  </DataTrigger>  <DataTrigger TargetType="Label" Binding="{Binding IsChecked}" Value="false">  <Setter Property="TextColor" Value="DarkGray"/>  </DataTrigger>  </Label.Triggers>  </Label>  <BoxView Grid.Row="1" Grid.ColumnSpan="2" HeightRequest="0.5" BackgroundColor="LightGray"/>  </Grid>  </DataTemplate>  </BindableLayout.ItemTemplate>  </StackLayout>  </ContentPage> |

Click the "Run" button to try it out.



### Note:

Bindable layouts should only be used when the collection of items to be displayed is small, and scrolling and selection aren't required. While scrolling can be provided by wrapping a bindable layout in a [ScrollView](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.scrollview), this is not recommended as bindable layouts lack UI virtualization. When scrolling is required, a scrollable view that includes UI virtualization, such as [ListView](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.listview) or [CollectionView](https://docs.microsoft.com/en-us/dotnet/api/xamarin.forms.collectionview), should be used. Failure to observe this recommendation can lead to performance issues.

I hope now you have understood what is Bindable Layout and how to use it in Xamarin.Forms.

Thanks for reading. Please share your comments and feedback. Happy Coding…!