MVVM Cross (v6)

[**Basic Binding And Dependency Injection**](#_sov4a7w6i7mt) **1**

[Core Project](#_8vndcob2usar) 1

[Android Project](#_zdeo8d2xtce3) 4

[iOS Project](#_r9drvl9lqy2j) 7

[**Value Converters**](#_g634fxwzrwl) **10**

[Core Project](#_gbf980kazwhy) 10

[Android Project](#_4aixqyiauzuc) 10

[iOS Project](#_9sorluallun2) 10

[**Listview Bindings**](#_q83n847jomxk) **11**

[Core Project](#_vvu9hovzen1z) 11

[Android Project](#_kzinb2cbg7s9) 13

[iOS Project](#_lxi2oy4b6qoi) 15

[**Show Images**](#_2778bvco0rrk) **17**

[Android Project](#_modm88fq9s2b) 17

[iOS Project](#_wv8melj9puoj) 18

# Basic Binding And Dependency Injection

## Core Project

1. Create a blank solution.
2. Add an standard library and call “**XXX.Core**”.
3. Delete **Class1**.
4. Add NuGet **MvvmCross**.
5. Add **Services** folder and add **ICalculationService** interface inside it.

public interface ICalculationService

{

decimal TipAmount(decimal subTotal, int generosity);

}

1. In the same folder add the interface implementation (**CalculationService**).

public class CalculationService : ICalculationService

{

public decimal TipAmount(decimal subTotal, int generosity)

{

var tip = subTotal \* (decimal)(generosity / 100.0);

return tip;

}

}

1. Add the folder **ViewModels** and the class **TipViewModel** inside it.

using System.Threading.Tasks;

using MvvmCross.ViewModels;

using Services;

public class TipViewModel : MvxViewModel

{

#region Attributes

private readonly ICalculationService calculationService;

private decimal subTotal;

private int generosity;

private decimal tip;

#endregion

#region Properties

public decimal SubTotal

{

get

{

return this.subTotal;

}

set

{

this.subTotal = value;

this.RaisePropertyChanged(() => this.SubTotal);

this.Recalculate();

}

}

public decimal Tip

{

get

{

return this.tip;

}

set

{

this.tip = value;

this.RaisePropertyChanged(() => this.Tip);

}

}

public int Generosity

{

get

{

return this.generosity;

}

set

{

this.generosity = value;

this.RaisePropertyChanged(() => this.Generosity);

this.Recalculate();

}

}

#endregion

#region Constructors

public TipViewModel(ICalculationService calculationService)

{

this.calculationService = calculationService;

}

#endregion

#region Methods

public override async Task Initialize()

{

await base.Initialize();

this.SubTotal = 100;

this.Generosity = 10;

this.Recalculate();

}

private void Recalculate()

{

this.Tip = this.calculationService.TipAmount(this.SubTotal, this.Generosity);

}

#endregion

}

1. In the root project add the **App** class.

using MvvmCross.IoC;

using MvvmCross.ViewModels;

using ViewModels;

public class App : MvxApplication

{

public override void Initialize()

{

this.CreatableTypes()

.EndingWith("Service")

.AsInterfaces()

.RegisterAsLazySingleton();

this.RegisterAppStart<TipViewModel>();

}

}

1. Congratulations you have ready the foundation for the solution.

## Android Project

1. Now add the android project and call **XXX.Droid**, use blank application template.
2. Add the reference to **Core** project and add the NuGet **MvvmCross**.
3. Add a reference to **Mono.Android.Export.dll**.
4. Delete the **MainActivity** activity and the **activity\_main** layout.
5. Into **Resources** folder, add the folder **drawable** and inside it add the files **Icon.png** and **splash.png**.
6. Into **layout** folder add the **SplashPage** layout.

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Loading...." />

</LinearLayout>

1. Into layout folder add the **TipPage** layout.

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="vertical"

android:padding="20dp"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textSize="24dp"

android:text="SubTotal" />

<EditText

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:inputType="number|numberDecimal"

android:textSize="24dp"

android:gravity="right"

local:MvxBind="Text SubTotal" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginTop="10dp"

android:textSize="24dp"

android:text="Generosity" />

<SeekBar

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:max="100"

local:MvxBind="Progress Generosity" />

<View

android:layout\_width="match\_parent"

android:layout\_height="1dp"

android:layout\_margin="30dp"

android:background="@android:color/darker\_gray" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textSize="24dp"

android:text="Tip to leave" />

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textColor="@android:color/holo\_blue\_dark"

android:textSize="24dp"

android:gravity="center"

local:MvxBind="Text Tip" />

</LinearLayout>

1. In **strings.xml** modify the application name.

<resources>

<string name="app\_name">MVVM Cross</string>

<string name="action\_settings">Settings</string>

</resources>

1. In values folder add the **SplashStyle.xml** file.

<?xml version="1.0" encoding="utf-8"?>

<resources>

<style name="Theme.Splash" parent="android:Theme">

<item name="android:windowBackground">@drawable/splash</item>

<item name="android:windowNoTitle">true</item>

</style>

</resources>

1. Add the folder **Views** and inside it add the class **SplashView**.

using Android.App;

using Android.Content.PM;

using Core;

using MvvmCross.Platforms.Android.Core;

using MvvmCross.Platforms.Android.Views;

[Activity(

Label = "@string/app\_name",

MainLauncher = true,

Icon = "@drawable/icon",

Theme = "@style/Theme.Splash",

NoHistory = true,

ScreenOrientation = ScreenOrientation.Portrait)]

public class SplashView : MvxSplashScreenActivity<MvxAndroidSetup<App>, App>

{

public SplashView() : base(Resource.Layout.SplashPage)

{

}

}

1. In the folder **Views** add **TipView**.

using Android.App;

using Android.OS;

using Core.ViewModels;

using MvvmCross.Platforms.Android.Views;

[Activity(Label = "@string/app\_name")]

public class TipView : MvxActivity<TipViewModel>

{

protected override void OnCreate(Bundle savedInstanceState)

{

base.OnCreate(savedInstanceState);

this.SetContentView(Resource.Layout.TipPage);

}

}

1. You’re ready to test the real cross project in android!

## iOS Project

1. Now let's do, something similar on iOS. Add the iOS project and call **XXX.iOS**, use blank application template.
2. Verify in Application properties the **Assembly name** and **Default namespace** are call correctly, I mean **XXX.iOS**.
3. Add the reference to **netstandard.dll**, browse to **C:\Windows\Microsoft.NET\Framework\v4.0.30319** to find it.
4. Add the reference to **Core** project and add the NuGet **MvvmCross**.
5. Modify the **AppDelegate** by:

using Core;

using Foundation;

using MvvmCross.Platforms.Ios.Core;

[Register("AppDelegate")]

public class AppDelegate : MvxApplicationDelegate<MvxIosSetup<App>, App>

{

}

1. Fix the name space in the **AppDelegate**.
2. Modify the **LaunchScreen.storyboard** by:



1. Add the folder **Views** and inside it add the view **HomeView.xib, HomeView.cs** and **HomeView.designer.cs**. And fix the namespaces.
2. Modify the view **HomeView.xib** similar to this:



1. Modify the class **HomeView**:

using Core.ViewModels;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Presenters.Attributes;

using MvvmCross.Platforms.Ios.Views;

[MvxRootPresentation(WrapInNavigationController = true)]

public partial class HomeView : MvxViewController<TipViewModel>

{

public override void ViewDidLoad()

{

base.ViewDidLoad();

var set = this.CreateBindingSet<HomeView, TipViewModel>();

set.Bind(this.AmountText).To(vm => vm.SubTotal);

set.Bind(this.GenerositySlider).To(vm => vm.Generosity);

set.Bind(this.TipLabel).To(vm => vm.Tip);

set.Apply();

}

}

1. You’re ready to test the project on iOS!

# Value Converters

## Core Project

1. Add the folder **Converters** and inside it, create the class: **DecimalToStringValueConverter**, it’s very important that the class name ends with **ValueConverter**.

using MvvmCross.Converters;

using System;

using System.Globalization;

public class DecimalToStringValueConverter : MvxValueConverter<decimal, string>

{

protected override string Convert(decimal value, Type targetType, object parameter, CultureInfo culture)

{

return $"{value:C2}";

}

}

## Android Project

1. Change the Page to call the converter:

<TextView

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textColor="@android:color/holo\_blue\_dark"

android:textSize="24dp"

android:gravity="center"

local:MvxBind="Text Tip,Converter=DecimalToString" />

1. Test it.

## iOS Project

1. Change the controller to call the converter:

set.Bind(this.TipLabel).To(vm => vm.Tip).WithConversion("DecimalToString");

1. Test it.

# Listview Bindings

## Core Project

1. In **Services**, add the **Kitten** class:

public class Kitten

{

public string Name { get; set; }

public int Price { get; set; }

public string ImageUrl { get; set; }

}

1. In **Services**, add the **IKittenGenesisService** interface:

public interface IKittenGenesisService

{

Kitten CreateNewKitten(string extra = "");

}

1. In **Services**, add the **KittenGenesisService** class:

using System;

using System.Collections.Generic;

public class KittenGenesisService : IKittenGenesisService

{

private readonly Random random = new Random();

private readonly List<string> names = new List<string>() {

"Tiddles",

"Amazon",

"Pepsi",

"Solomon",

"Butler",

"Snoopy",

"Harry",

"Holly",

"Paws",

"Polly",

"Dixie",

"Fern",

"Cousteau",

"Frankenstein",

"Bazooka",

"Casanova",

"Fudge",

"Comet" };

public Kitten CreateNewKitten(string extra = "")

{

var rnd = Random(20) + 300;

return new Kitten()

{

Name = $"{this.names[this.Random(this.names.Count)]}{extra}",

ImageUrl = $"http://placekitten.com/{rnd}/{rnd}",

Price = this.RandomPrice()

};

}

protected int Random(int count)

{

return this.random.Next(count);

}

protected int RandomPrice()

{

return this.Random(23) + 3;

}

}

1. In **ViewModels**, add the **KittensViewModel** class:

using Core.Services;

using MvvmCross.ViewModels;

using System.Collections.Generic;

public class KittensViewModel : MvxViewModel

{

private List<Kitten> kittens;

public List<Kitten> Kittens

{

get => this.kittens;

set => this.SetProperty(ref this.kittens, value);

}

public KittensViewModel(IKittenGenesisService service)

{

var newList = new List<Kitten>();

for (int i = 0; i < 100; i++)

{

var newKitten = service.CreateNewKitten(i.ToString());

newList.Add(newKitten);

}

this.Kittens = newList;

}

}

1. Change the **App** class, to start with the new View Model:

using MvvmCross.IoC;

using MvvmCross.ViewModels;

using ViewModels;

public class App : MvxApplication

{

public override void Initialize()

{

this.CreatableTypes()

.EndingWith("Service")

.AsInterfaces()

.RegisterAsLazySingleton();

//this.RegisterAppStart<TipViewModel>();

this.RegisterAppStart<FirstViewModel>();

}

}

1. That's all in the core project.

## Android Project

1. In **layout**, add the **Item\_Kitten** layout:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<LinearLayout

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text Name" />

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="20dp"

local:MvxBind="Text Price" />

</LinearLayout>

</LinearLayout>

1. In **layout**, add the **KittensPage** layout:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<MvxListView

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

local:MvxBind="ItemsSource Kittens"

local:MvxItemTemplate="@layout/item\_kitten"/>

</LinearLayout>

1. In **Views**, add the **KittensView** class:

using Android.App;

using Android.OS;

using MvvmCross.Platforms.Android.Views;

using Core.ViewModels;

[Activity(Label = "@string/app\_name")]

public class KittensView : MvxActivity<KittensViewModel>

{

protected override void OnCreate(Bundle bundle)

{

base.OnCreate(bundle);

this.SetContentView(Resource.Layout.KittensPage);

}

}

1. Test it in Android.

## iOS Project

1. In **Views**, add the **KittensViewController** class:

using Core.ViewModels;

using Foundation;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Binding.Views;

using MvvmCross.Platforms.Ios.Views;

[Register("FirstViewController")]

public class FirstViewController using Core.ViewModels;

using Foundation;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Binding.Views;

using MvvmCross.Platforms.Ios.Views;

[Register("KittensViewController")]

public class KittensViewController : MvxTableViewController<KittensViewModel>

{

public override void ViewDidLoad()

{

base.ViewDidLoad();

var source = new MvxStandardTableViewSource(TableView, "TitleText Name;ImageUrl ImageUrl");

TableView.Source = source;

var set = this.CreateBindingSet<KittensViewController, KittensViewModel>();

set.Bind(source).To(vm => vm.Kittens);

set.Apply();

TableView.ReloadData();

}

}

1. Test it in iOS.
2. Now we personalize the table view. In **Views** add a **Table View Cell**, call, **KittenCell**. Then modify the **KittenCell.xib** by:



1. Modify the **KittenCell.cs** by:

using System;

using Foundation;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Binding.Views;

using Core.Services;

using UIKit;

public partial class KittenCell : MvxTableViewCell

{

public static readonly NSString Key = new NSString("KittenCell");

public static readonly UINib Nib;

static KittenCell()

{

Nib = UINib.FromName("KittenCell", NSBundle.MainBundle);

}

protected KittenCell(IntPtr handle) : base(handle)

{

this.DelayBind(() =>

{

var set = this.CreateBindingSet<KittenCell, Kitten>();

set.Bind(this.NameLabel).To(vm => vm.Name);

set.Bind(this.PriceLabel).To(vm => vm.Price);

set.Apply();

});

}

}

1. Then modify the **KittensViewController.cs** by:

base.ViewDidLoad();

//var source = new MvxStandardTableViewSource(TableView, "TitleText Name;ImageUrl ImageUrl");

var source = new MvxSimpleTableViewSource(TableView, KittenCell.Key, KittenCell.Key);

TableView.RowHeight = 70;

TableView.Source = source;

var set = this.CreateBindingSet<KittensViewController, KittensViewModel>();

1. Test it.

# Show Images

## Android Project

1. Add the NuGet **Xamarin.FFImageLoading**.
2. Modify the **Item\_Kitten**:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<ffimageloading.cross.MvxCachedImageView

android:layout\_width="75dp"

android:layout\_height="75dp"

android:layout\_margin="10dp"

local:MvxBind="ImagePath ImageUrl" />

<LinearLayout

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text Name" />

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="20dp"

local:MvxBind="Text Price" />

</LinearLayout>

</LinearLayout>

1. Add permission to INTERNET in the android manifest:

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0" package="MVVMCross5.Droid.MVVMCross5.Droid" android:installLocation="auto">

<uses-sdk android:minSdkVersion="21" android:targetSdkVersion="27" />

<uses-permission android:name="android.permission.INTERNET" />

<application android:allowBackup="true" android:icon="@mipmap/ic\_launcher" android:label="@string/app\_name" android:roundIcon="@mipmap/ic\_launcher\_round" android:supportsRtl="true" android:theme="@style/AppTheme"></application>

</manifest>

1. Test it.

## iOS Project

1. Add the NuGet **Xamarin.FFImageLoading**.
2. In folder **Views** and **CustomTableViewCell**:

using System;

using System.Drawing;

using FFImageLoading.Cross;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Binding.Views;

using MVVMCross6.Core.Services;

using UIKit;

public partial class CustomTableViewCell : MvxTableViewCell

{

public CustomTableViewCell(IntPtr handle) : base(handle)

{

var imageControl = new MvxCachedImageView(new RectangleF(10, 10, 80, 80));

this.Add(imageControl);

var namelLabel = new UILabel(new RectangleF(110, 10, 200, 40));

this.Add(namelLabel);

var priceLabel = new UILabel(new RectangleF(110, 60, 200, 40));

this.Add(priceLabel);

this.DelayBind(

() =>

{

var set = this.CreateBindingSet<CustomTableViewCell, Kitten>();

set.Bind(namelLabel).To(vm => vm.Name);

set.Bind(priceLabel).To(vm => vm.Price);

set.Bind(imageControl).For(vm => vm.ImagePath).To(vm => vm.ImageUrl);

set.Apply();

}

);

}

}

1. Modify the **KittensViewController** class:

using Core.ViewModels;

using Foundation;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Binding.Views;

using MvvmCross.Platforms.Ios.Views;

[Register("KittensViewController")]

public class KittensViewController : MvxTableViewController<KittensViewModel>

{

public override void ViewDidLoad()

{

base.ViewDidLoad();

var source = new MvxSimpleTableViewSource(TableView, typeof(CustomTableViewCell), nameof(CustomTableViewCell));

TableView.RowHeight = 100;

TableView.Source = source;

var set = this.CreateBindingSet<KittensViewController, KittensViewModel>();

set.Bind(source).To(vm => vm.Kittens);

set.Apply();

TableView.ReloadData();

}

}

1. Test it.

# Multi Page

## Core Project

1. In **ViewModels** add **ThreeViewModel**:

using MvvmCross.ViewModels;

public class ThreeViewModel : MvxViewModel

{

private string theAnswer;

public string TheAnswer

{

get { return this.theAnswer; }

set

{

this.theAnswer = value;

this.RaisePropertyChanged(() => this.TheAnswer);

}

}

public ThreeViewModel()

{

this.theAnswer = "Fuck yeah!";

}

}

1. In **ViewModels** add **TwoViewModel**:

using System.Windows.Input;

using MvvmCross.Commands;

using MvvmCross.Navigation;

using MvvmCross.ViewModels;

public class TwoViewModel : MvxViewModel

{

private string name = "Second";

private MvxCommand goThirdCommand;

private readonly IMvxNavigationService navigationService;

public string Name

{

get { return this.name; }

set

{

this.name = value;

this.RaisePropertyChanged(() => this.Name);

}

}

public ICommand GoThirdCommand

{

get

{

this.goThirdCommand = this.goThirdCommand ?? new MvxCommand(this.DoGoThird);

return this.goThirdCommand;

}

}

public TwoViewModel(IMvxNavigationService navigationService)

{

this.navigationService = navigationService;

}

private async void DoGoThird()

{

await this.navigationService.Navigate<ThreeViewModel>();

}

}

1. In **ViewModels** add **OneViewModel**:

using MvvmCross.Commands;

using MvvmCross.Navigation;

using MvvmCross.ViewModels;

using System.Windows.Input;

public class OneViewModel : MvxViewModel

{

private string hello = "Hello MvvmCross";

private MvxCommand myCommand;

private MvxCommand goSecondCommand;

private readonly IMvxNavigationService navigationService;

public string Hello

{

get { return this.hello; }

set

{

this.hello = value;

this.RaisePropertyChanged(() => this.Hello);

}

}

public ICommand MyCommand

{

get

{

this.myCommand = this.myCommand ?? new MvxCommand(this.DoMyCommand);

return this.myCommand;

}

}

public ICommand GoSecondCommand

{

get

{

this.goSecondCommand = this.goSecondCommand ?? new MvxCommand(this.DoGoSecond);

return this.goSecondCommand;

}

}

public OneViewModel(IMvxNavigationService navigationService)

{

this.navigationService = navigationService;

}

private async void DoGoSecond()

{

await this.navigationService.Navigate<TwoViewModel>();

}

private void DoMyCommand()

{

this.Hello += " World";

}

}

1. Change the **App** class to start with the new view model:

//this.RegisterAppStart<TipViewModel>();

//this.RegisterAppStart<KittensViewModel>();

this.RegisterAppStart<OneViewModel>();

## Android Project

1. In **layout** add **OnePage**:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<EditText

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text Hello" />

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text Hello" />

<Button

android:text="Click Me"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

local:MvxBind="Click MyCommand" />

<Button

android:text="Go to Second"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

local:MvxBind="Click GoSecondCommand" />

</LinearLayout>

1. In **layout** add **TwoPage**:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="vertical"

android:background="#800000"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<EditText

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text Name" />

<Button

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

android:text="go third"

local:MvxBind="Click GoThirdCommand" />

</LinearLayout>

1. In **layout** add **ThreePage**:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="vertical"

android:background="#800000"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text TheAnswer" />

</LinearLayout>

1. In **Views** add **OneView**:

using Android.App;

using Android.OS;

using Core.ViewModels;

using MvvmCross.Platforms.Android.Views;

[Activity(Label = "@string/app\_name")]

public class OneView : MvxActivity<OneViewModel>

{

protected override void OnCreate(Bundle bundle)

{

base.OnCreate(bundle);

this.SetContentView(Resource.Layout.OnePage);

}

}

1. In **Views** add **TwoView**:

using Android.App;

using Android.OS;

using Core.ViewModels;

using MvvmCross.Platforms.Android.Views;

[Activity(Label = "@string/app\_name")]

public class TwoView : MvxActivity<TwoViewModel>

{

protected override void OnCreate(Bundle bundle)

{

base.OnCreate(bundle);

this.SetContentView(Resource.Layout.TwoPage);

}

}

1. In **Views** add **ThreeView**:

using Android.App;

using Android.OS;

using Core.ViewModels;

using MvvmCross.Platforms.Android.Views;

[Activity(Label = "@string/app\_name")]

public class ThreeView : MvxActivity<ThreeViewModel>

{

protected override void OnCreate(Bundle bundle)

{

base.OnCreate(bundle);

this.SetContentView(Resource.Layout.ThreePage);

}

}

1. Test it.

## iOS Project

1. In **Views** add **OneView**:

using System.Drawing;

using Foundation;

using Core.ViewModels;

using UIKit;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Views;

[Register("OneView")]

public class OneView : MvxViewController<OneViewModel>

{

public override void ViewDidLoad()

{

View = new UIView() { BackgroundColor = UIColor.White };

base.ViewDidLoad();

var label = new UILabel(new RectangleF(10, 10, 300, 40));

Add(label);

var textField = new UITextField(new RectangleF(10, 50, 300, 40));

Add(textField);

var button = new UIButton(UIButtonType.RoundedRect);

button.SetTitle("Click Me", UIControlState.Normal);

button.Frame = new RectangleF(10, 90, 300, 40);

Add(button);

var button2 = new UIButton(UIButtonType.RoundedRect);

button2.SetTitle("Go Second", UIControlState.Normal);

button2.Frame = new RectangleF(10, 130, 300, 40);

Add(button2);

var set = this.CreateBindingSet<OneView, OneViewModel>();

set.Bind(label).To(vm => vm.Hello);

set.Bind(textField).To(vm => vm.Hello);

set.Bind(button).To(vm => vm.MyCommand);

set.Bind(button2).To(vm => vm.GoSecondCommand);

set.Apply();

}

}

1. In **Views** add **TwoView**:

using System.Drawing;

using Core.ViewModels;

using Foundation;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Views;

using UIKit;

[Register("TwoView")]

public class TwoView : MvxViewController<TwoViewModel>

{

public override void ViewDidLoad()

{

View = new UIView() { BackgroundColor = UIColor.Red };

base.ViewDidLoad();

var label = new UILabel(new RectangleF(10, 10, 300, 40));

Add(label);

var button = new UIButton(UIButtonType.RoundedRect);

button.SetTitle("Go Third", UIControlState.Normal);

button.Frame = new RectangleF(10, 50, 300, 40);

Add(button);

var set = this.CreateBindingSet<TwoView, TwoViewModel>();

set.Bind(label).To(vm => vm.Name);

set.Bind(button).To(vm => vm.GoThirdCommand);

set.Apply();

}

}

1. In **Views** add **ThreeView**:

using System.Drawing;

using Foundation;

using MvvmCross.Binding.BindingContext;

using MvvmCross.iOS.Views;

using MVVMCross5.Core.ViewModels;

using UIKit;

[Register("ThreeView")]

public class ThreeView : MvxViewController<ThreeViewModel>

{

public override void ViewDidLoad()

{

View = new UIView() { BackgroundColor = UIColor.Cyan };

base.ViewDidLoad();

var label = new UILabel(new RectangleF(10, 10, 300, 40));

Add(label);

var set = this.CreateBindingSet<ThreeView, ThreeViewModel>();

set.Bind(label).To(vm => vm.TheAnswer);

set.Apply();

}

}

1. Test it.

# Consuming Rest Services

## Core Project

1. Add NuGet **MvvmCross.Plugin.Json**.
2. In folder **Services** add the class **ImageLinks**:

using Newtonsoft.Json;

public class ImageLinks

{

[JsonProperty(PropertyName = "smallThumbnail")]

public string SmallThumbnail { get; set; }

[JsonProperty(PropertyName = "thumbnail")]

public string Thumbnail { get; set; }

}

1. In the same folder add the class **VolumeInfo**:

using System.Collections.Generic;

using Newtonsoft.Json;

public class VolumeInfo

{

[JsonProperty(PropertyName = "title")]

public string Title { get; set; }

[JsonProperty(PropertyName = "authors")]

public List<string> Authors { get; set; }

public string AuthorSummary

{

get { return this.Authors == null ? "-" : string.Join(", ", this.Authors); }

}

[JsonProperty(PropertyName = "publisher")]

public string Publisher { get; set; }

[JsonProperty(PropertyName = "publishedDate")]

public string PublishedDate { get; set; }

[JsonProperty(PropertyName = "descrption")]

public string Descrption { get; set; }

[JsonProperty(PropertyName = "pageCount")]

public int PageCount { get; set; }

[JsonProperty(PropertyName = "averageRating")]

public double AverageRating { get; set; }

[JsonProperty(PropertyName = "ratingsCount")]

public int RatingsCount { get; set; }

[JsonProperty(PropertyName = "imageLinks")]

public ImageLinks ImageLinks { get; set; }

[JsonProperty(PropertyName = "language")]

public string Language { get; set; }

[JsonProperty(PropertyName = "previewLink")]

public string PreviewLink { get; set; }

[JsonProperty(PropertyName = "infoLink")]

public string InfoLink { get; set; }

[JsonProperty(PropertyName = "canonicalVolumeLink")]

public string CanonicalVolumeLink { get; set; }

}

1. In the same folder add the class **BookSearchItem**:

using Newtonsoft.Json;

public class BookSearchItem

{

[JsonProperty(PropertyName = "kind")]

public string Kind { get; set; }

[JsonProperty(PropertyName = "id")]

public string Id { get; set; }

[JsonProperty(PropertyName = "volumeInfo")]

public VolumeInfo VolumeInfo { get; set; }

public override string ToString()

{

return this.VolumeInfo.Title;

}

}

1. In the same folder add the class **BookSearchResult**:

using System.Collections.Generic;

using Newtonsoft.Json;

public class BookSearchResult

{

[JsonProperty(PropertyName = "items")]

public List<BookSearchItem> Items { get; set; }

}

1. In the same folder add the class **ISimpleRestService**:

using System;

public interface ISimpleRestService

{

void MakeRequest<T>(string requestUrl, string verb, Action<T> successAction, Action<Exception> errorAction);

}

1. In the same folder add the class **SimpleRestService**:

using System;

using System.IO;

using System.Net;

using MvvmCross;

using MvvmCross.Base;

public class SimpleRestService : ISimpleRestService

{

public void MakeRequest<T>(string requestUrl, string verb, Action<T> successAction, Action<Exception> errorAction)

{

var request = (HttpWebRequest)WebRequest.Create(requestUrl);

request.Method = verb;

request.Accept = "application/json";

this.MakeRequest(

request,

(response) =>

{

if (successAction != null)

{

T toReturn;

try

{

toReturn = this.Deserialize<T>(response);

}

catch (Exception ex)

{

errorAction(ex);

return;

}

successAction(toReturn);

}

},

(error) =>

{

if (errorAction != null)

{

errorAction(error);

}

}

);

}

private void MakeRequest(HttpWebRequest request, Action<string> successAction, Action<Exception> errorAction)

{

request.BeginGetResponse(token =>

{

try

{

using (var response = request.EndGetResponse(token))

{

using (var stream = response.GetResponseStream())

{

var reader = new StreamReader(stream);

successAction(reader.ReadToEnd());

}

}

}

catch (WebException ex)

{

errorAction(ex);

}

}, null);

}

private T Deserialize<T>(string responseBody)

{

var serializer = Mvx.Resolve<IMvxJsonConverter>();

var toReturn = serializer.DeserializeObject<T>(responseBody);

return toReturn;

}

}

1. In the same folder add the class **IBooksService**:

using System;

public interface IBooksService

{

void StartSearchAsync(string whatFor, Action<BookSearchResult> success, Action<Exception> error);

}

1. In the same folder add the class **BooksService**:

using System;

public class BooksService : IBooksService

{

private readonly ISimpleRestService simpleRestService;

public BooksService(ISimpleRestService simpleRestService)

{

this.simpleRestService = simpleRestService;

}

public void StartSearchAsync(string whatFor, Action<BookSearchResult> success, Action<Exception> error)

{

var address = $"https://www.googleapis.com/books/v1/volumes?q={Uri.EscapeDataString(whatFor)}";

this.simpleRestService.MakeRequest<BookSearchResult>(address, "GET", success, error);

}

}

1. In the folder **ViewModels** add the class **BooksViewModel**:

using System.Collections.Generic;

using MvvmCross.Core.ViewModels;

using Services;

public class BooksViewModel : MvxViewModel

{

private readonly IBooksService books;

public BooksViewModel(IBooksService books)

{

this.books = books;

}

private string searchTerm;

public string SearchTerm

{

get { return this.searchTerm; }

set

{

this.searchTerm = value;

this.RaisePropertyChanged(() => this.SearchTerm);

this.Update();

}

}

private List<BookSearchItem> results;

public List<BookSearchItem> Results

{

get { return this.results; }

set

{

this.results = value;

this.RaisePropertyChanged(() => this.Results);

}

}

private void Update()

{

this.books.StartSearchAsync(SearchTerm,

result => Results = result.Items,

error => { });

}

}

1. Modify the **App** class to start up with **BooksViewModel**:

//this.RegisterAppStart<TipViewModel>();

//this.RegisterAppStart<FirstViewModel>();

//this.RegisterAppStart<OneViewModel>();

this.RegisterAppStart<BooksViewModel>();

## Android Project

1. In **layout** add **Item\_Book** layout:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<ffimageloading.cross.MvxCachedImageView

android:layout\_width="100dp"

android:layout\_height="100dp"

local:MvxBind="ImagePath VolumeInfo.ImageLinks.SmallThumbnail"

android:layout\_margin="5dp" />

<LinearLayout

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="30dp"

android:text="Title here"

local:MvxBind="Text VolumeInfo.Title"

android:layout\_margin="10dp" />

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="Author here"

android:textSize="20dp"

local:MvxBind="Text VolumeInfo.AuthorSummary"

android:layout\_marginLeft="20dp" />

</LinearLayout>

</LinearLayout>

1. In **layout** add **BooksPage** layout:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<EditText

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text SearchTerm" />

<MvxListView

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

local:MvxBind="ItemsSource Results"

local:MvxItemTemplate="@layout/item\_book"/>

</LinearLayout>

1. In **Views** add **BooksView** class:

using Android.App;

using Android.OS;

using Core.ViewModels;

using MvvmCross.Platforms.Android.Views;

[Activity(Label = "@string/app\_name")]

public class BooksView : MvxActivity<BooksViewModel>

{

protected override void OnCreate(Bundle bundle)

{

base.OnCreate(bundle);

this.SetContentView(Resource.Layout.BooksPage);

}

}

## iOS Project

1. In **Views** add the **BookTableViewCell** class:

using System;

using System.Drawing;

using FFImageLoading.Cross;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Binding.Views;

using MVVMCross6.Core.Services;

using UIKit;

public class BookTableViewCell : MvxTableViewCell

{

public BookTableViewCell(IntPtr handle) : base(handle)

{

var imageControl = new MvxCachedImageView(new RectangleF(10, 10, 80, 80));

this.Add(imageControl);

var namelLabel = new UILabel(new RectangleF(110, 10, 200, 40));

this.Add(namelLabel);

var authorsLabel = new UILabel(new RectangleF(110, 60, 200, 40));

this.Add(authorsLabel);

this.DelayBind(

() =>

{

var set = this.CreateBindingSet<BookTableViewCell, BookSearchItem>();

set.Bind(namelLabel).To(vm => vm.VolumeInfo.Title);

set.Bind(authorsLabel).To(vm => vm.VolumeInfo.AuthorSummary);

set.Bind(imageControl).For(vm => vm.ImagePath).To(vm => vm.VolumeInfo.ImageLinks.SmallThumbnail);

set.Apply();

}

);

}

}

1. In **Views** add the **BooksView** class:

using System.Drawing;

using Core.ViewModels;

using Foundation;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Binding.Views;

using MvvmCross.Platforms.Ios.Views;

using UIKit;

[Register("BooksView")]

public class BooksView : MvxViewController<BooksViewModel>

{

public override void ViewDidLoad()

{

View = new UIView() { BackgroundColor = UIColor.White };

base.ViewDidLoad();

var textField = new UITextField(new RectangleF(10, 70, 300, 40));

Add(textField);

var tableView = new UITableView(new RectangleF(0, 110, 320, 440), UITableViewStyle.Plain);

Add(tableView);

tableView.RowHeight = 100;

var source = new MvxSimpleTableViewSource(tableView, typeof(BookTableViewCell), nameof(BookTableViewCell));

tableView.Source = source;

var set = this.CreateBindingSet<BooksView, BooksViewModel>();

set.Bind(textField).To(vm => vm.SearchTerm);

set.Bind(source).To(vm => vm.Results);

set.Apply();

tableView.ReloadData();

}

}

1. Test it.

## Improve Data Loading

1. In **Views** modify the **BooksViewModel** class:

using System;

using System.Collections.Generic;

using System.Threading;

using MvvmCross.ViewModels;

using Services;

public class BooksViewModel : MvxViewModel

{

private readonly IBooksService books;

private readonly object lockObject = new object();

private Timer timer;

private string searchTerm;

private List<BookSearchItem> results;

private bool isLoading;

public bool IsLoading

{

get { return this.isLoading; }

set

{

this.isLoading = value;

this.RaisePropertyChanged(() => this.IsLoading);

}

}

public List<BookSearchItem> Results

{

get { return this.results; }

set

{

this.results = value;

this.RaisePropertyChanged(() => this.Results);

}

}

public string SearchTerm

{

get { return this.searchTerm; }

set

{

this.searchTerm = value;

this.RaisePropertyChanged(() => this.SearchTerm);

this.ScheduleUpdate();

}

}

public BooksViewModel(IBooksService books)

{

this.books = books;

}

private void ScheduleUpdate()

{

lock (this.lockObject)

{

if (this.timer == null)

{

this.timer = new Timer(this.OnTimerTick, null, TimeSpan.FromSeconds(1.0), TimeSpan.Zero);

}

else

{

this.timer.Change(TimeSpan.FromSeconds(1.0), TimeSpan.Zero);

}

}

}

private void OnTimerTick(object state)

{

lock (lockObject)

{

this.timer.Dispose();

this.timer = null;

}

this.Update();

}

private void Update()

{

this.IsLoading = true;

this.books.StartSearchAsync(this.SearchTerm,

result =>

{

this.IsLoading = false;

this.Results = result.Items;

},

error =>

{

this.IsLoading = false;

});

}

}

# Locations

## Core Project

1. Add NuGet **MvvmCross.Plugin.Location** and **MvvmCross.Plugin.Messenger** to the whole mobile projects and **MvvmCross.Plugin.Location.Fused** in Android.
2. In **Views** add **LocationViewModel**:

using MvvmCross.Plugin.Location;

using MvvmCross.ViewModels;

public class LocationViewModel : MvxViewModel

{

private readonly IMvxLocationWatcher watcher;

private double longitude;

private double latitude;

public double Longitude

{

get { return this.longitude; }

set

{

this.longitude = value;

this.RaisePropertyChanged(() => this.Longitude); }

}

public double Latitude

{

get { return this.latitude; }

set

{

this.latitude = value;

this.RaisePropertyChanged(() => this.Latitude); }

}

public LocationViewModel(IMvxLocationWatcher watcher)

{

this.watcher = watcher;

this.watcher.Start(new MvxLocationOptions(), this.OnLocation, this.OnError);

}

private void OnLocation(MvxGeoLocation location)

{

Latitude = location.Coordinates.Latitude;

Longitude = location.Coordinates.Longitude;

}

private void OnError(MvxLocationError error)

{

var message = error.ToString();

}

}

1. Modify the **App** to start with the new ViewModel:

this.RegisterAppStart<LocationViewModel>();

## Android Project

1. Add the layout **LocationPage**:

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:local="http://schemas.android.com/apk/res-auto"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

android:text="The lat and lng are:" />

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text Latitude" />

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:textSize="40dp"

local:MvxBind="Text Longitude" />

</LinearLayout>

1. In **Views** add the **LocationView** class.

using Android.App;

using Android.OS;

using Core.ViewModels;

using MvvmCross.Platforms.Android.Views;

[Activity(Label = "@string/app\_name")]

public class LocationView : MvxActivity<LocationViewModel>

{

protected override void OnCreate(Bundle bundle)

{

base.OnCreate(bundle);

this.SetContentView(Resource.Layout.LocationPage);

}

}

1. Add this permissions on Android manifest:

<uses-permission android:name="android.permission.INTERNET" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

1. Test it.

## iOS Project

1. In **Views** add the **LocationView** class:

using System.Drawing;

using Foundation;

using MvvmCross.Binding.BindingContext;

using MvvmCross.Platforms.Ios.Views;

using Core.ViewModels;

using UIKit;

[Register("LocationView")]

public class LocationView : MvxViewController<LocationViewModel>

{

public override void ViewDidLoad()

{

this.View = new UIView() { BackgroundColor = UIColor.White };

base.ViewDidLoad();

var latitudeLabel = new UILabel(new RectangleF(10, 60, 300, 40));

this.Add(latitudeLabel);

var longitudeLabel = new UITextField(new RectangleF(10, 100, 300, 40));

this.Add(longitudeLabel);

var set = this.CreateBindingSet<LocationView, LocationViewModel>();

set.Bind(latitudeLabel).To(vm => vm.Latitude);

set.Bind(longitudeLabel).To(vm => vm.Longitude);

set.Apply();

}

}

1. Test it.