#MonkeyConf

Fabulous Functional Frontends

Mark Allibone

Mobile Lead

Rey Technology

@mallibone

#MonkeyConf

SPONSORS

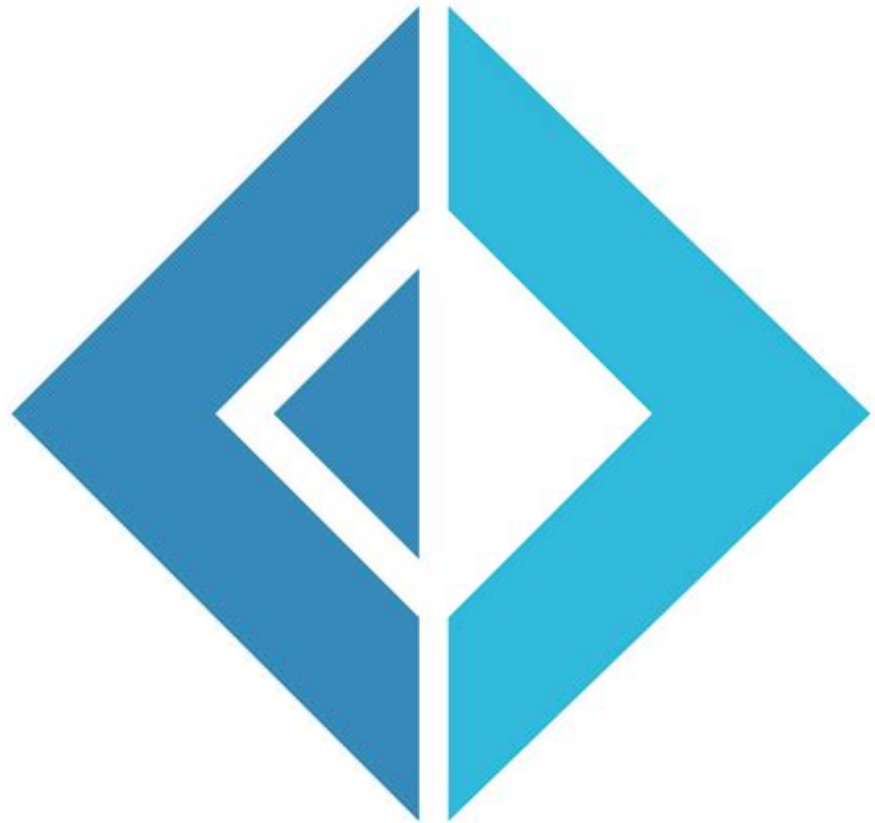
plain concepts

devs dna

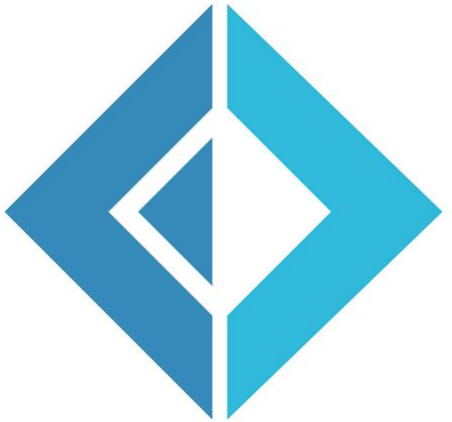
Beezy

NAMOCODE

UXDIVERS
PRODUCT DESIGN STUDIO

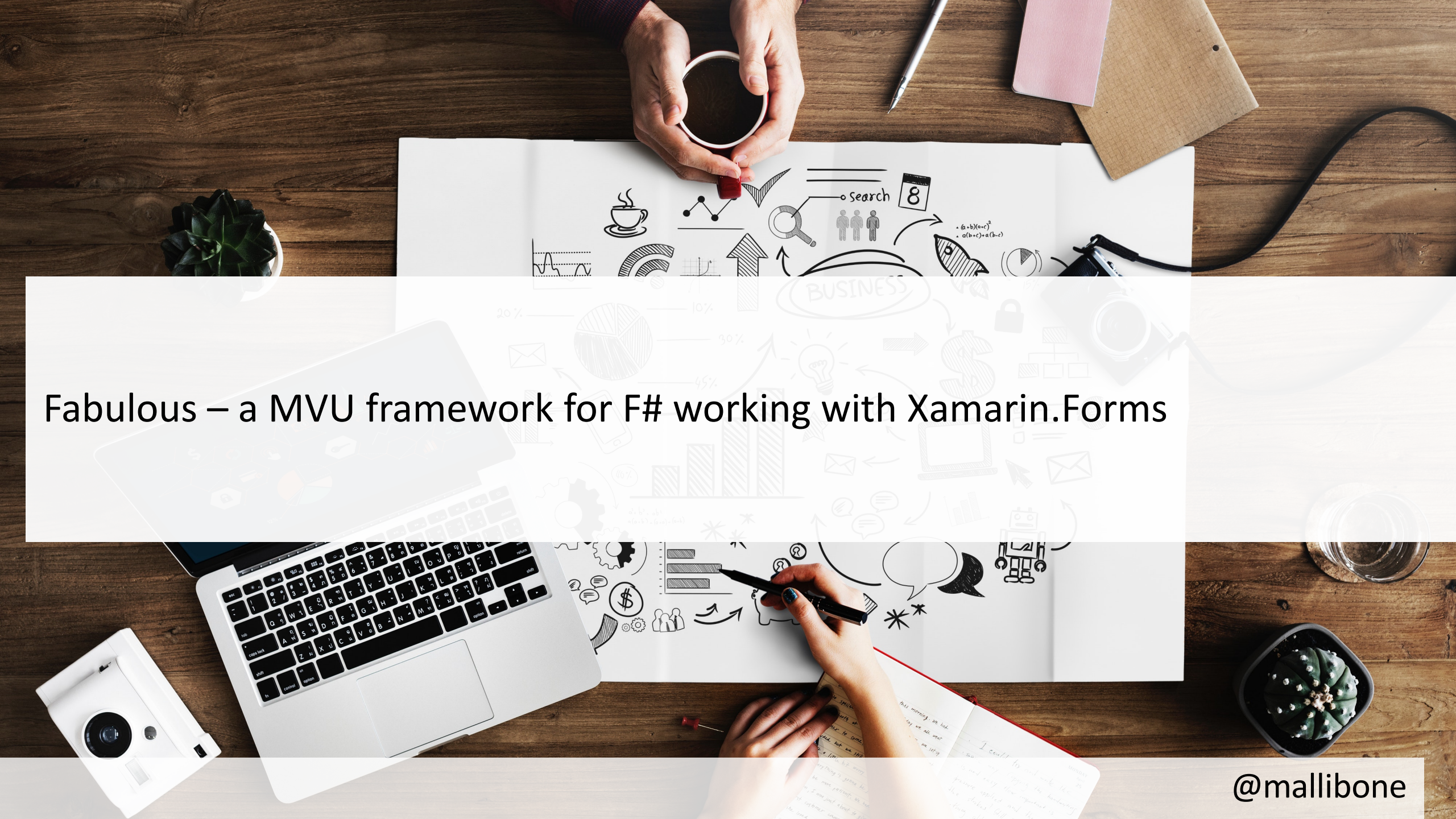


- No Nulls
- Immutability
- Declarative Programming
- Based on .Net



F# is great for BLOBs, FLOBs – overall a great general-purpose language

@mallibone

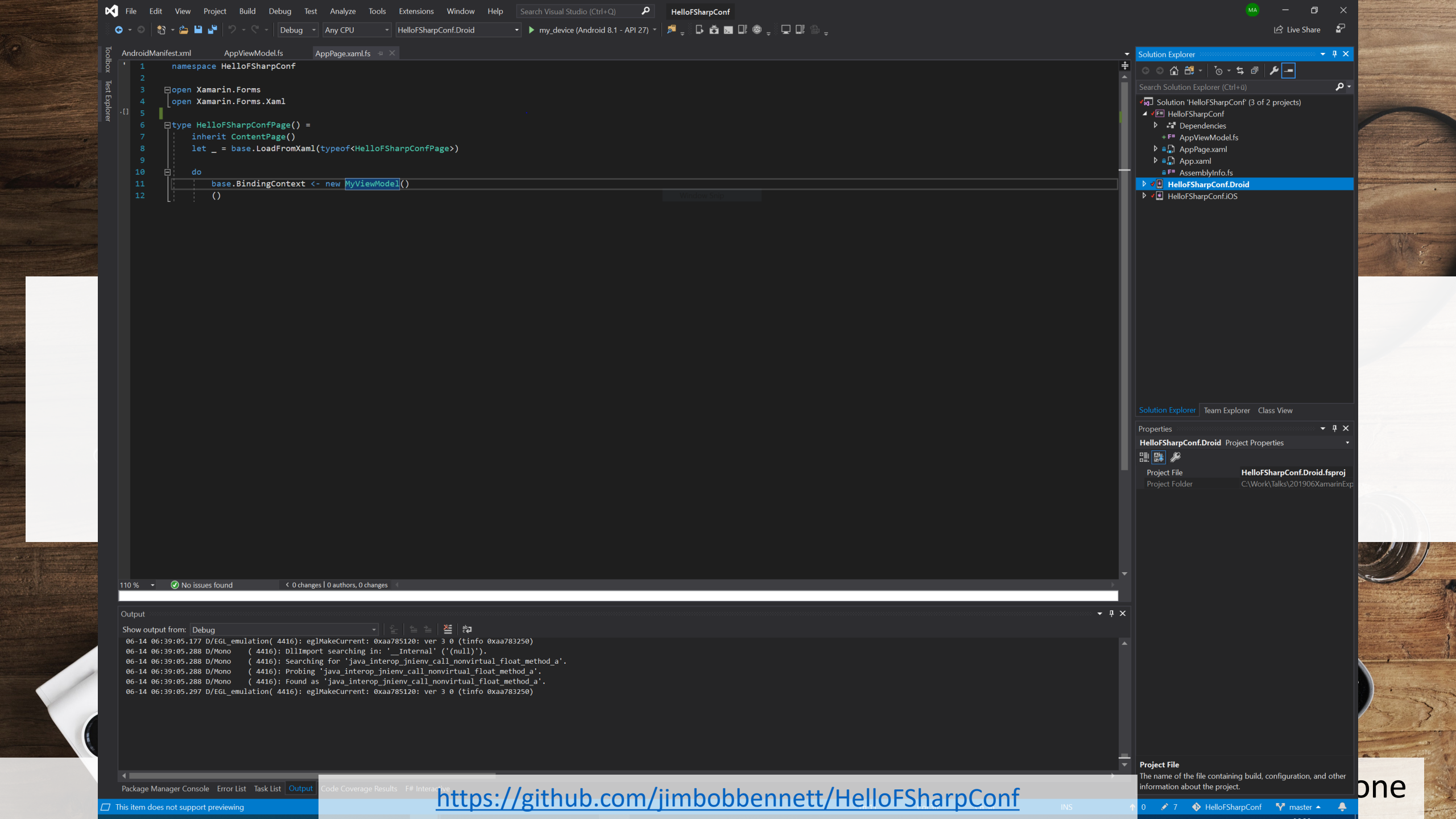


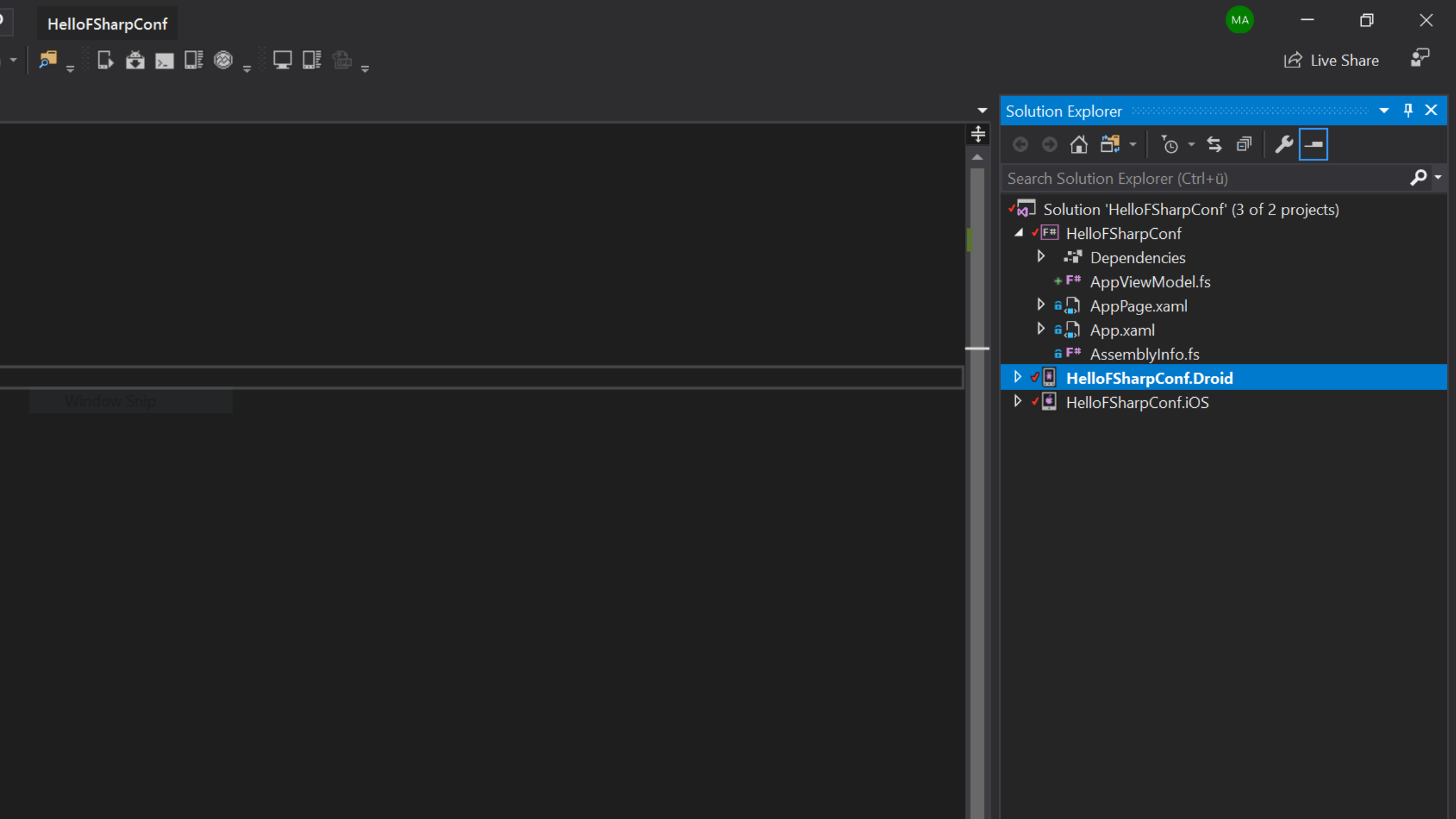
Fabulous – a MVU framework for F# working with Xamarin.Forms

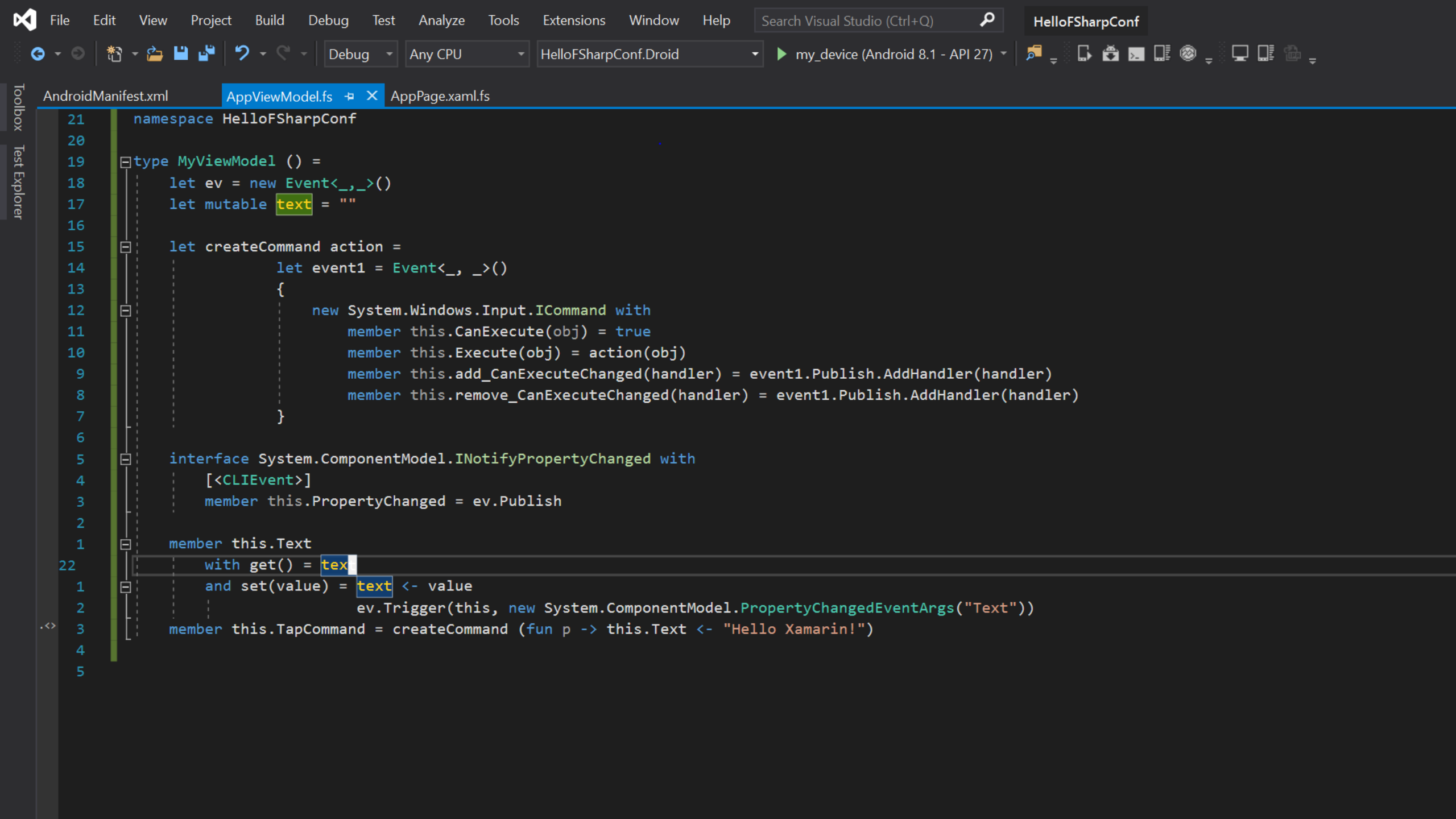
@mallibone

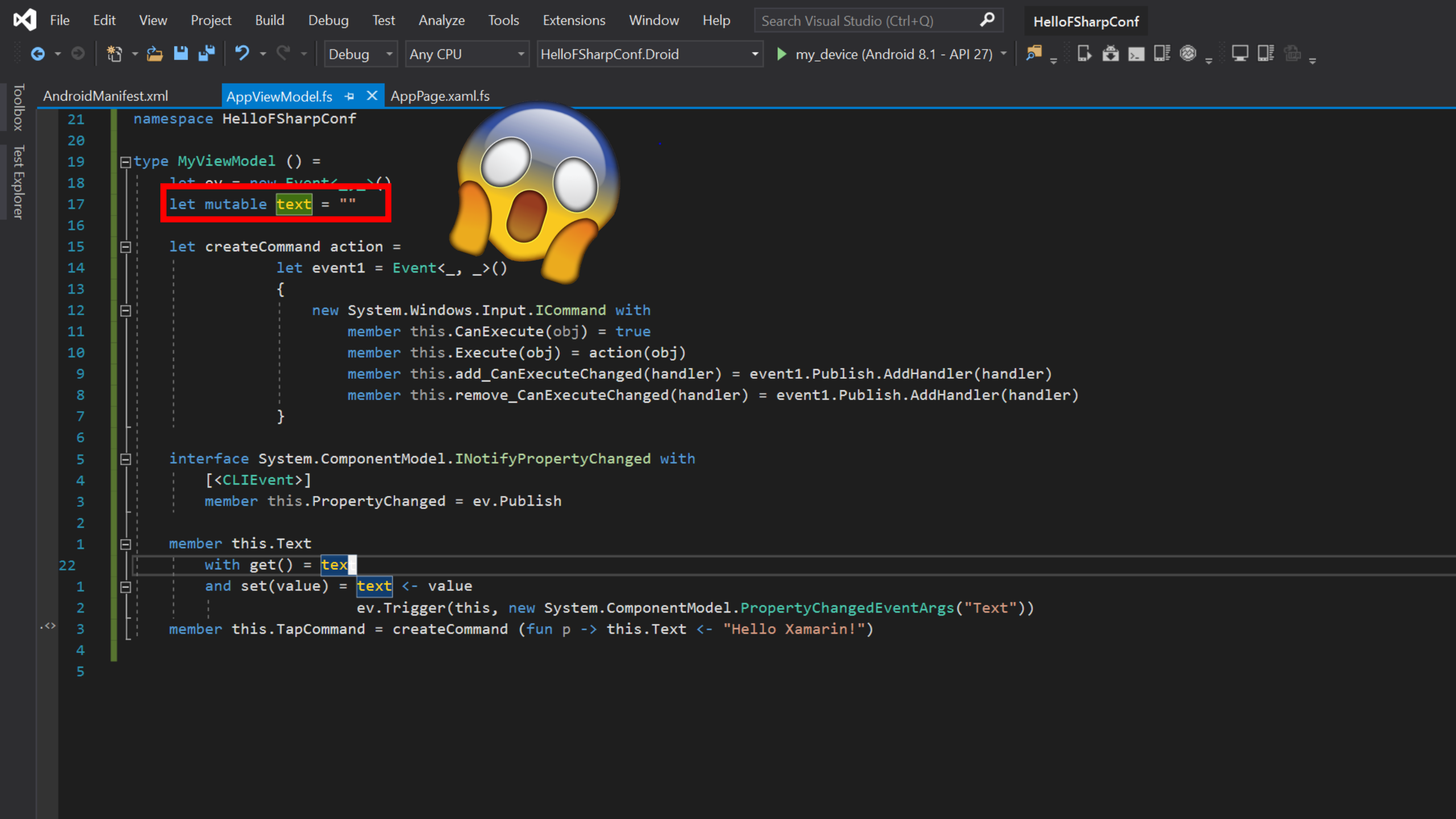
Why not just Xamarin Forms?!

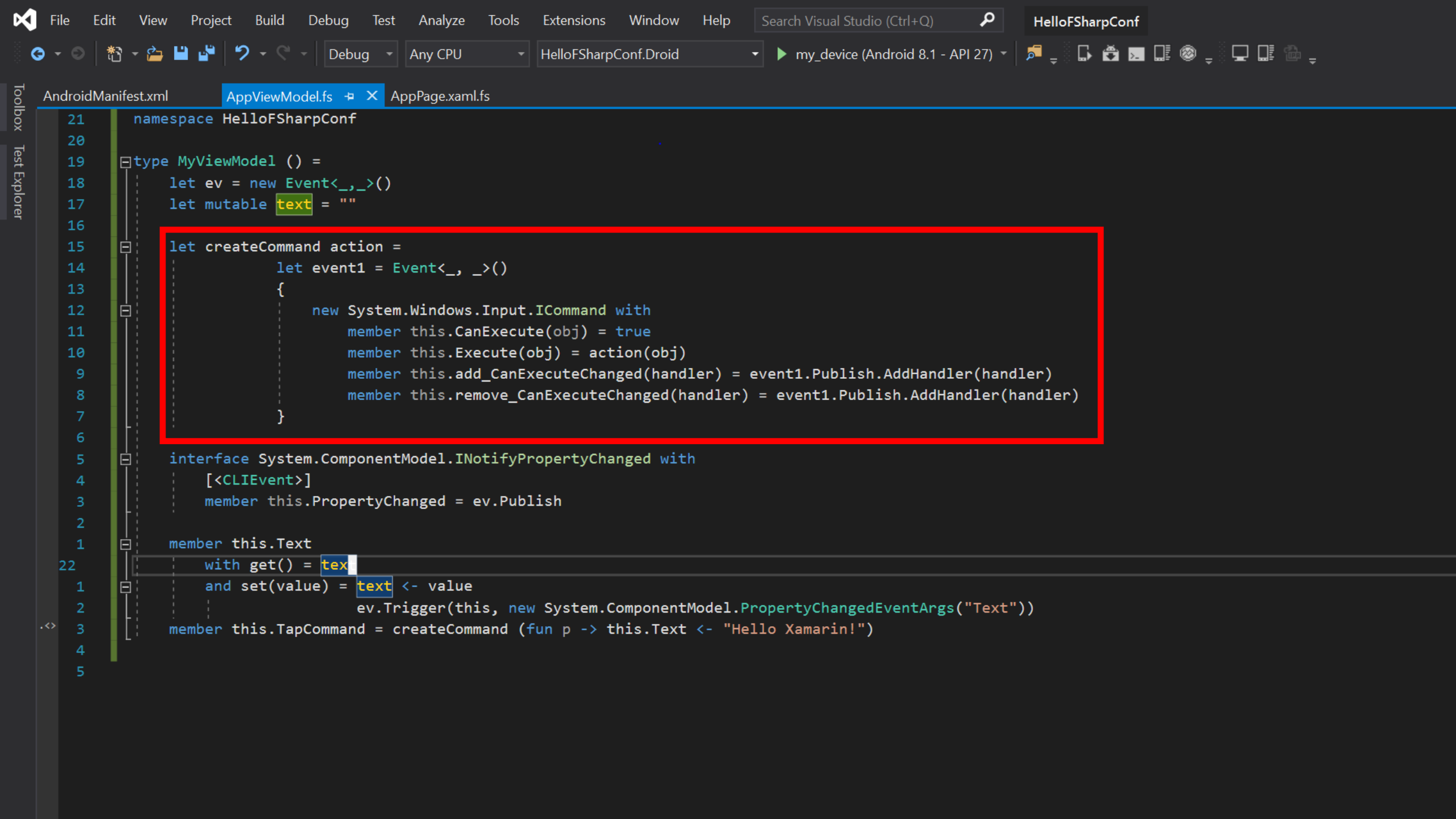
@mallibone

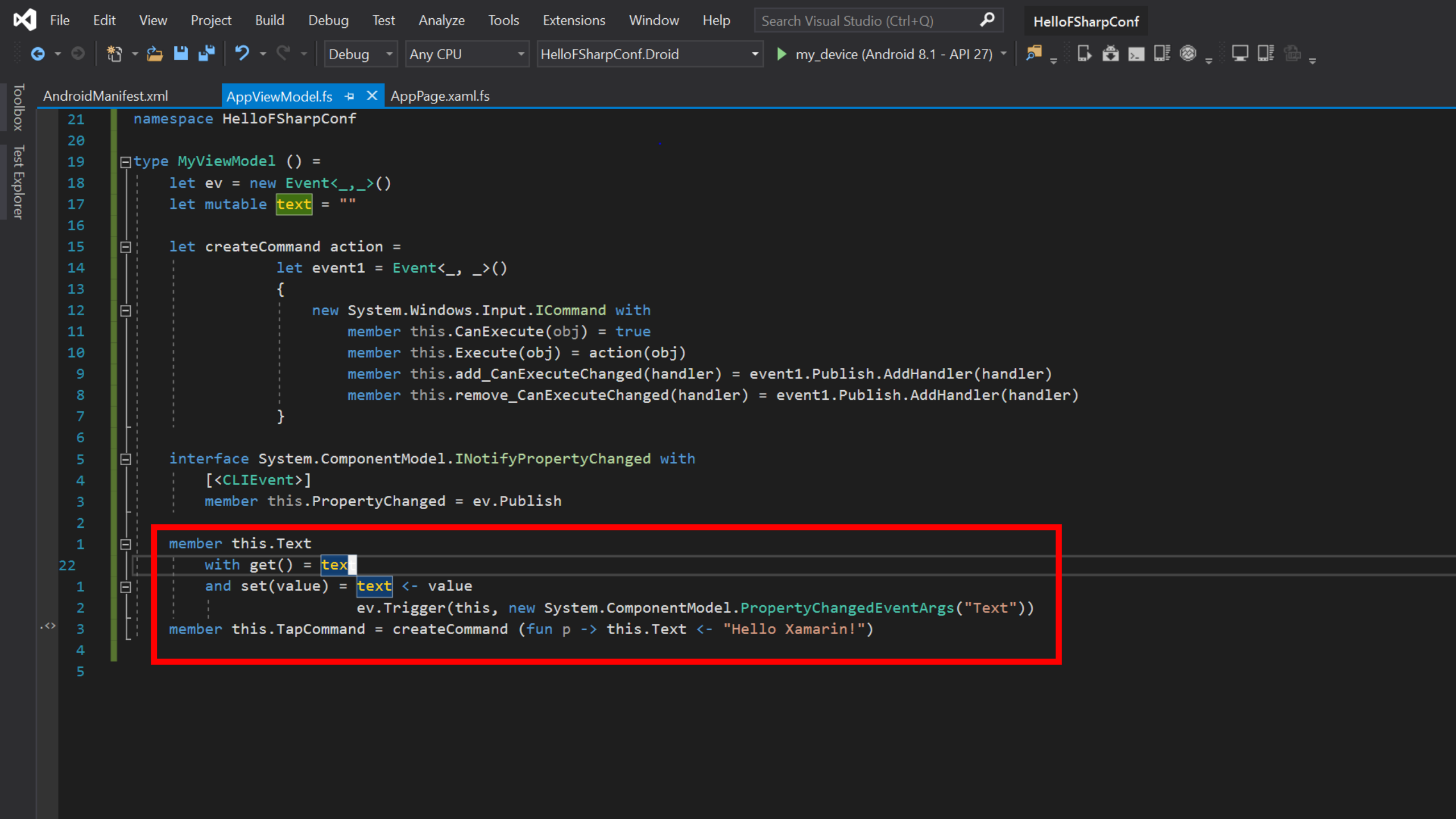

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
HelloFSharpConf
Debug
Any CPU
HelloFSharpConf.Droid
my_device (Android 8.1 - API 27)
Live Share
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
namespace HelloFSharpConf
open Xamarin.Forms
open Xamarin.Forms.Xaml
type HelloFSharpConfPage() =
inherit ContentPage()
let _ = base.LoadFromXaml(typeof<HelloFSharpConfPage>)
do
base.BindingContext <- new MyViewModel()
()
110 %
No issues found
0 changes | 0 authors, 0 changes
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'HelloFSharpConf' (3 of 2 projects)
HelloFSharpConf
Dependencies
AppViewModel.fs
AppPage.xaml
App.xaml
AssemblyInfo.fs
HelloFSharpConf.Droid
HelloFSharpConf.iOS
Solution Explorer
Team Explorer
Class View
Properties
HelloFSharpConf.Droid Project Properties
Project File
HelloFSharpConf.Droid.fsproj
Project Folder
C:\Work\Talks\201906XamarinExp
Output
Show output from: Debug
06-14 06:39:05.177 D/EGL_emulation(4416): eglMakeCurrent: 0xaa785120: ver 3 0 (tinfo 0xaa783250)
06-14 06:39:05.288 D/Mono (4416): DllImport searching in: '__Internal' ('(null)').
06-14 06:39:05.288 D/Mono (4416): Searching for 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.288 D/Mono (4416): Probing 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.288 D/Mono (4416): Found as 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.297 D/EGL_emulation(4416): eglMakeCurrent: 0xaa785120: ver 3 0 (tinfo 0xaa783250)
Package Manager Console
Error List
Task List
Output
Code Coverage Results
F# Interactive
Project File
The name of the file containing build, configuration, and other information about the project.
This item does not support previewing
https://github.com/jimbobbennett/HelloFSharpConf
HelloFSharpConf
master

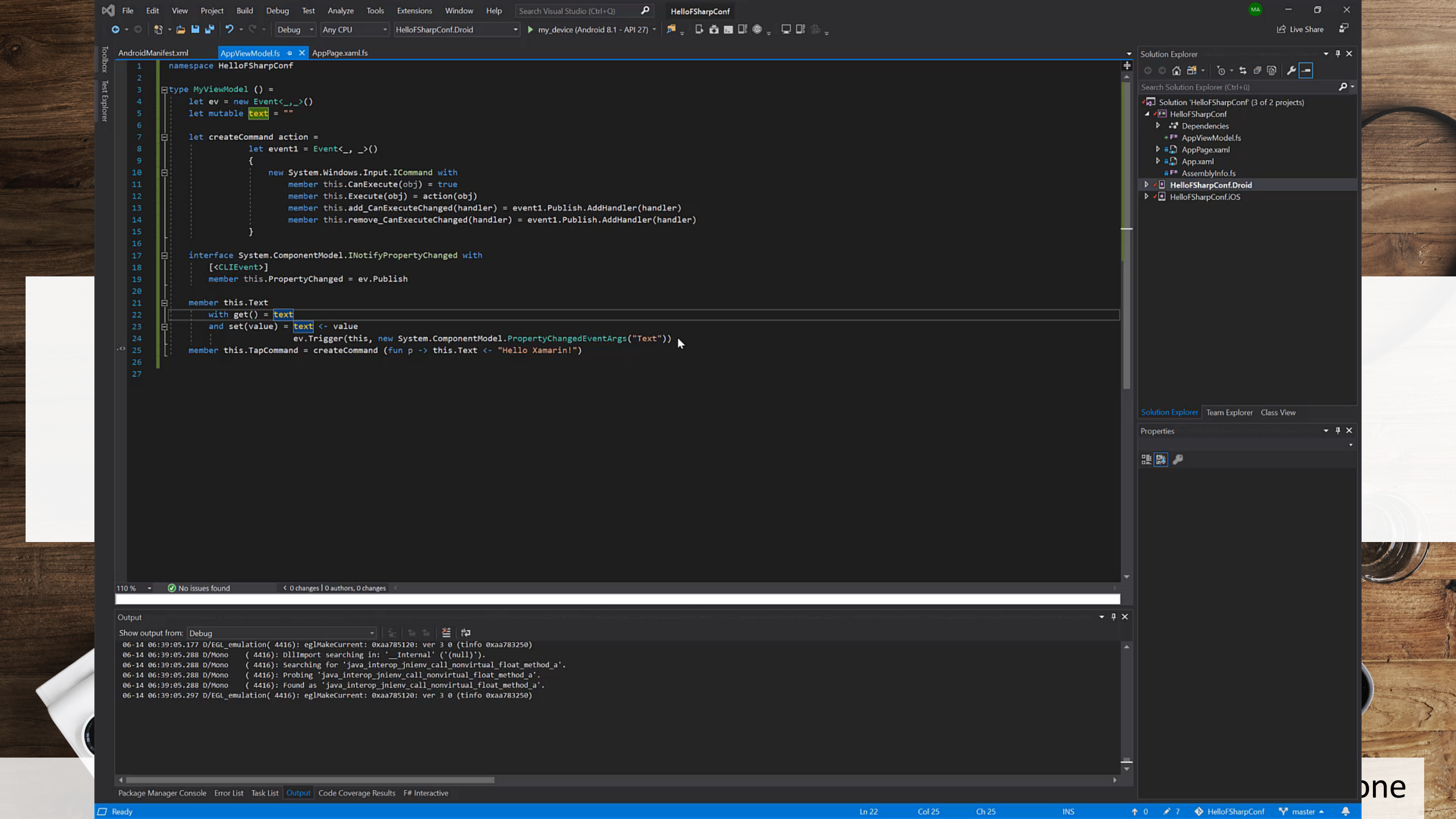

HelloFSharpConf
MA
Live Share
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'HelloFSharpConf' (3 of 2 projects)
HelloFSharpConf
Dependencies
AppViewModel.fs
AppPage.xaml
App.xaml
AssemblyInfo.fs
HelloFSharpConf.Droid
HelloFSharpConf.iOS


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
HelloFSharpConf
Debug
Any CPU
HelloFSharpConf.Droid
my_device (Android 8.1 - API 27)
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
Toolbox
Test Explorer
namespace HelloFSharpConf
type MyViewModel () =
let ev = new Event<_,_>()
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = tex
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
HelloFSharpConf
Debug
Any CPU
HelloFSharpConf.Droid
my_device (Android 8.1 - API 27)
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
Toolbox
Test Explorer
namespace HelloFSharpConf
type MyViewModel () =
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = tex
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
HelloFSharpConf
Debug
Any CPU
HelloFSharpConf.Droid
my_device (Android 8.1 - API 27)
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
Toolbox
Test Explorer
namespace HelloFSharpConf
type MyViewModel () =
let ev = new Event<_,_>()
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = tex
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
HelloFSharpConf
Debug
Any CPU
HelloFSharpConf.Droid
my_device (Android 8.1 - API 27)
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
Toolbox
Test Explorer
namespace HelloFSharpConf
type MyViewModel () =
let ev = new Event<_,_>()
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = tex
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")

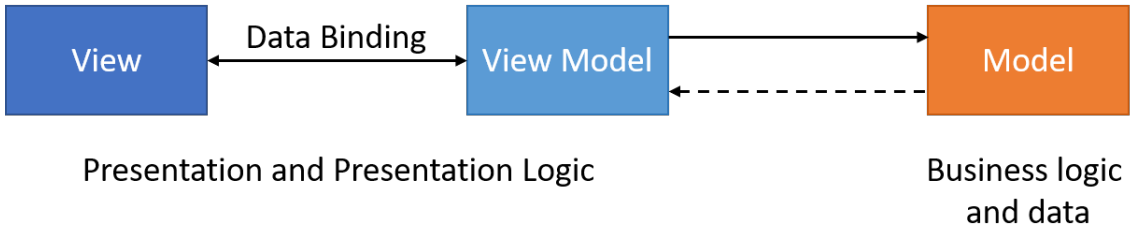

File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
HelloFSharpConf
Debug
Any CPU
HelloFSharpConf.Droid
my_device (Android 8.1 - API 27)
Live Share
AndroidManifest.xml
AppViewModel.fs
AppPage.xaml.fs
namespace HelloFSharpConf
type MyViewModel () =
let ev = new Event<_,_>()
let mutable text = ""
let createCommand action =
let event1 = Event<_, _>()
{
new System.Windows.Input.ICommand with
member this.CanExecute(obj) = true
member this.Execute(obj) = action(obj)
member this.add_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
member this.remove_CanExecuteChanged(handler) = event1.Publish.AddHandler(handler)
}
interface System.ComponentModel.INotifyPropertyChanged with
[<CLIEvent>]
member this.PropertyChanged = ev.Publish
member this.Text
with get() = text
and set(value) = text <- value
ev.Trigger(this, new System.ComponentModel.PropertyChangedEventArgs("Text"))
member this.TapCommand = createCommand (fun p -> this.Text <- "Hello Xamarin!")
110 %
No issues found
0 changes | 0 authors, 0 changes
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'HelloFSharpConf' (3 of 2 projects)
HelloFSharpConf
Dependencies
AppViewModel.fs
AppPage.xaml
App.xaml
AssemblyInfo.fs
HelloFSharpConf.Droid
HelloFSharpConf.iOS
Solution Explorer
Team Explorer
Class View
Properties
Output
Show output from: Debug
06-14 06:39:05.177 D/EGL_emulation(4416): eglMakeCurrent: 0xaa785120: ver 3 0 (tinfo 0xaa783250)
06-14 06:39:05.288 D/Mono (4416): DllImport searching in: '__Internal' ('(null)').
06-14 06:39:05.288 D/Mono (4416): Searching for 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.288 D/Mono (4416): Probing 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.288 D/Mono (4416): Found as 'java_interop_jnienv_call_nonvirtual_float_method_a'.
06-14 06:39:05.297 D/EGL_emulation(4416): eglMakeCurrent: 0xaa785120: ver 3 0 (tinfo 0xaa783250)
Package Manager Console
Error List
Task List
Output
Code Coverage Results
F# Interactive
Ready
Ln 22
Col 25
Ch 25
INS
HelloFSharpConf
master

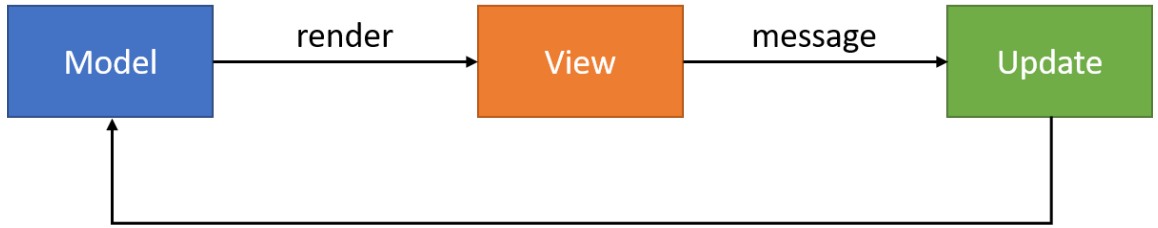
F# can do OO programming but it is functional first

MVU the functional MV* View Pattern

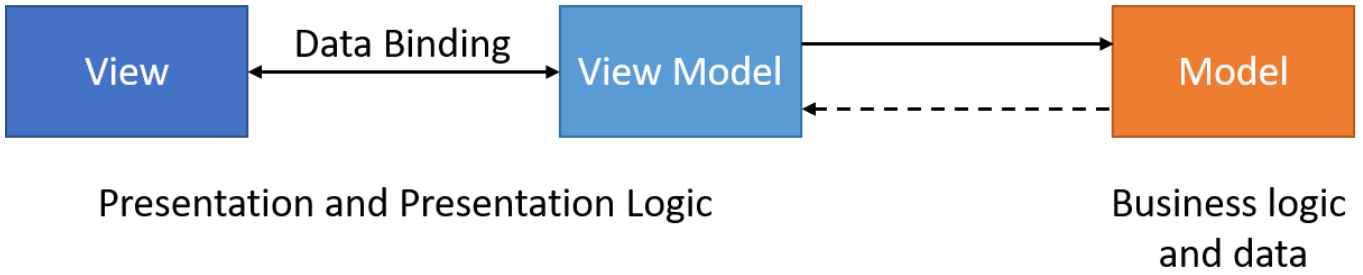
@mallibone

MVVM vs MVU

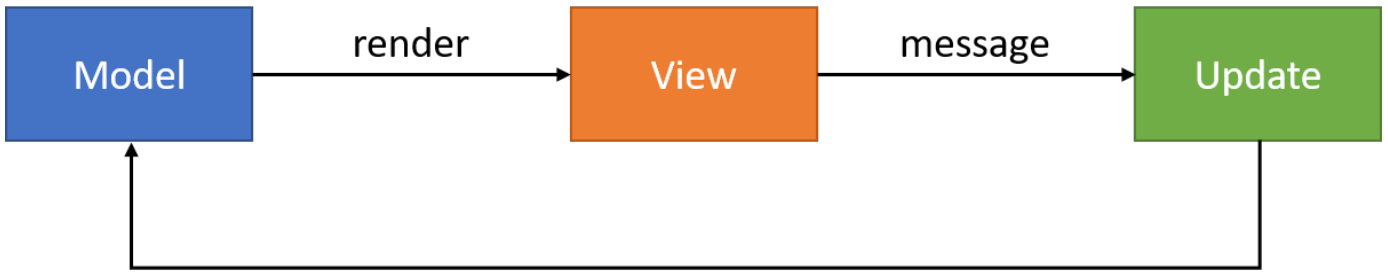

View
Data Binding
View Model
Model
Presentation and Presentation Logic
Business logic
and data


Model
render
View
message
Update

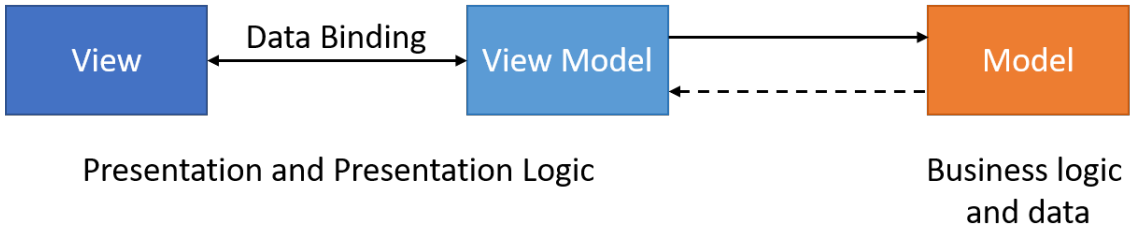
Model View View Model

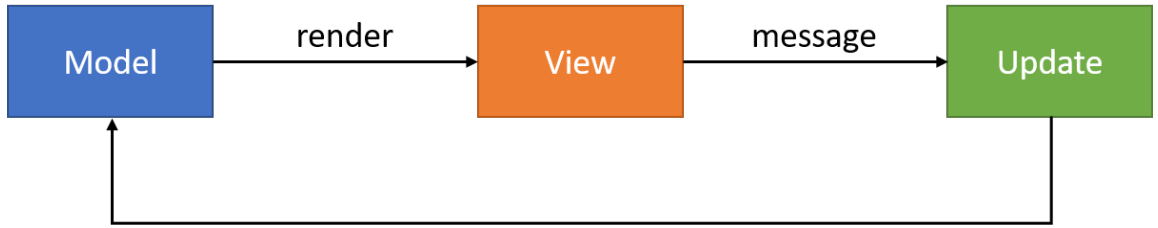


Model View Update



MVVM vs MVU


View
Data Binding
View Model
Model
Presentation and Presentation Logic
Business logic
and data


Model
render
View
message
Update

Functional Frontends

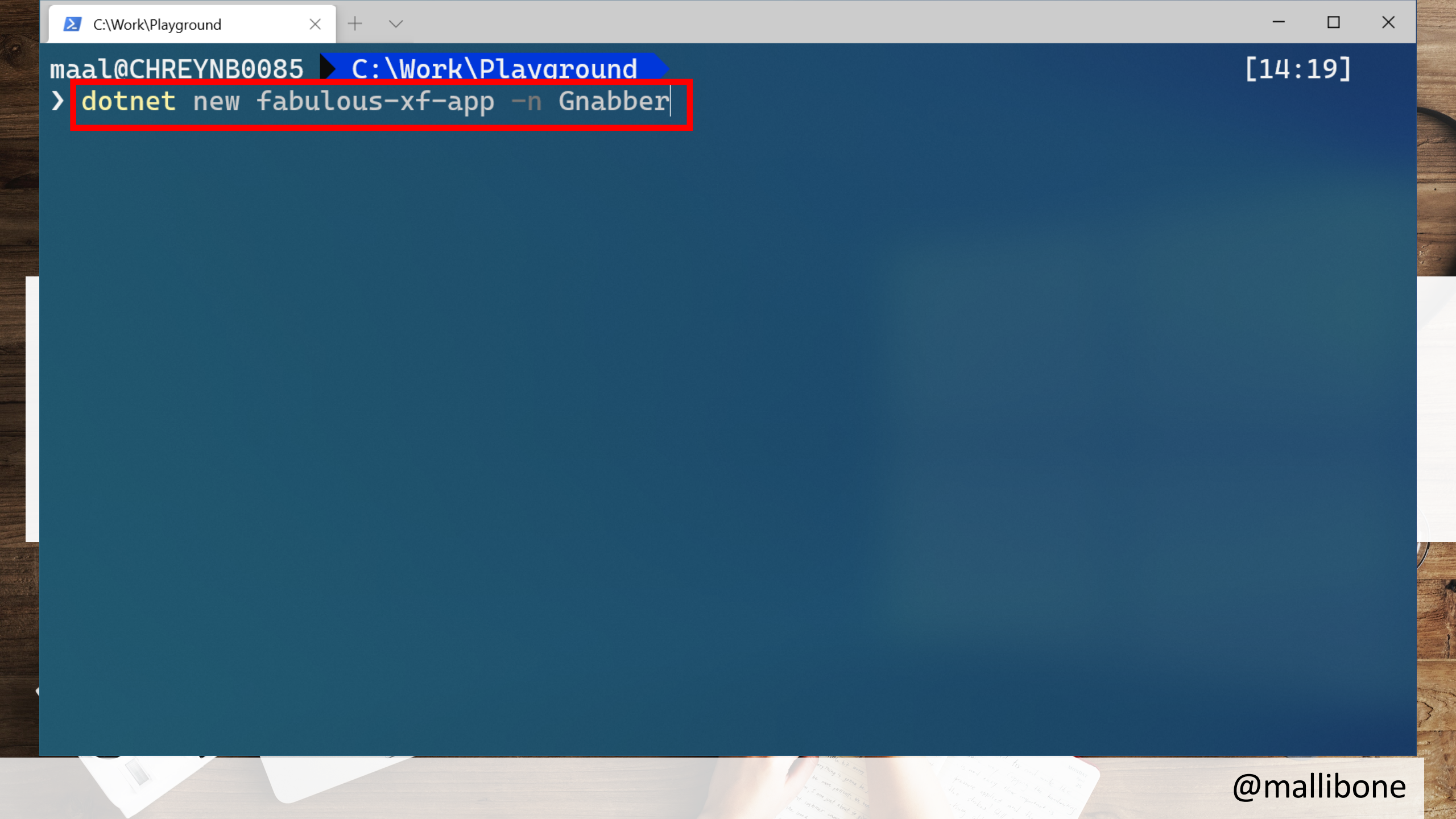
Getting started

1. Install Visual Studio or Visual Studio for Mac and enable both Xamarin and .NET Core support, these are listed as 'Mobile development with .NET' and '.NET Core Cross-platform development' respectively.

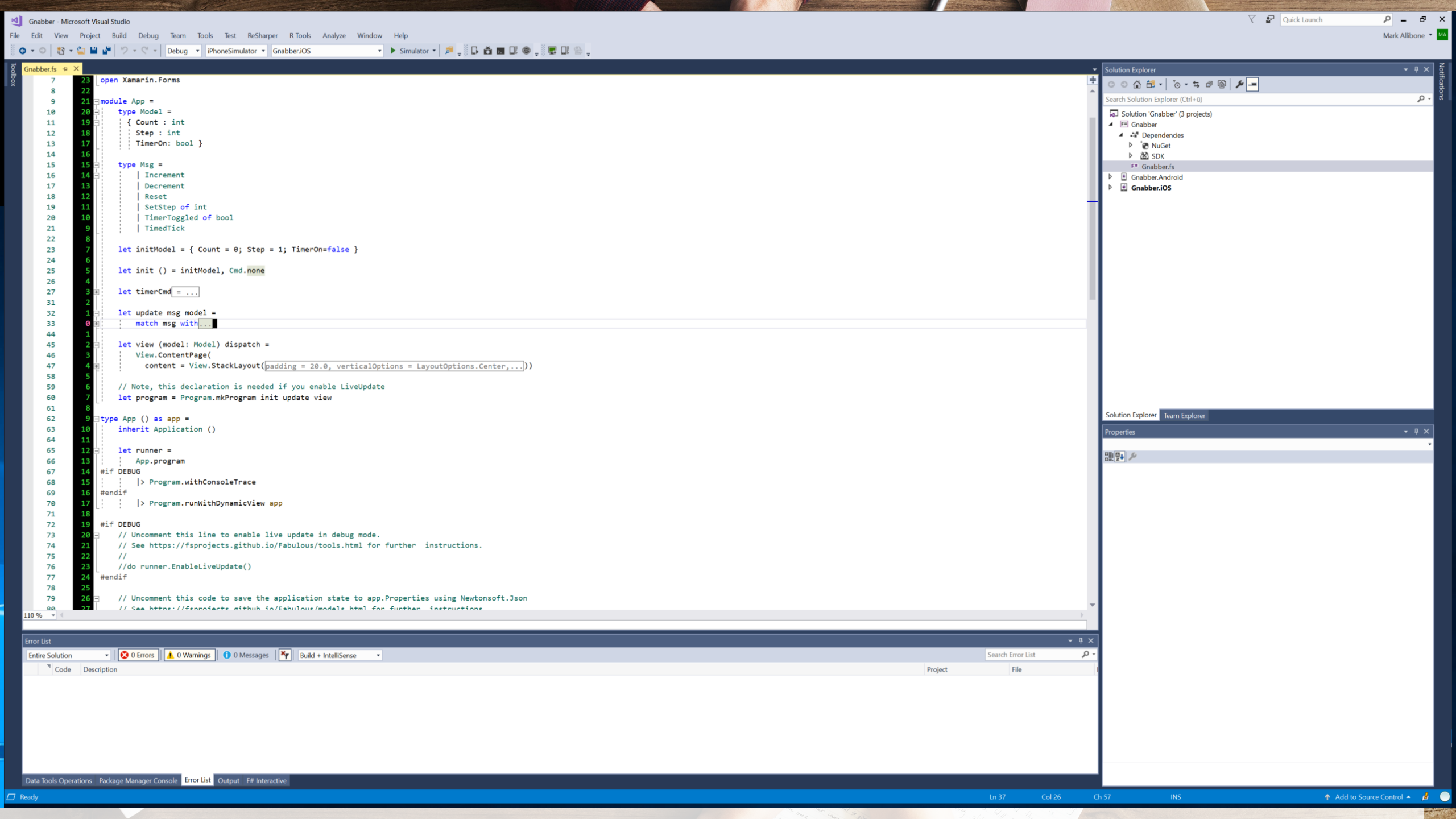
2. Open a command prompt window and install the template pack by entering:

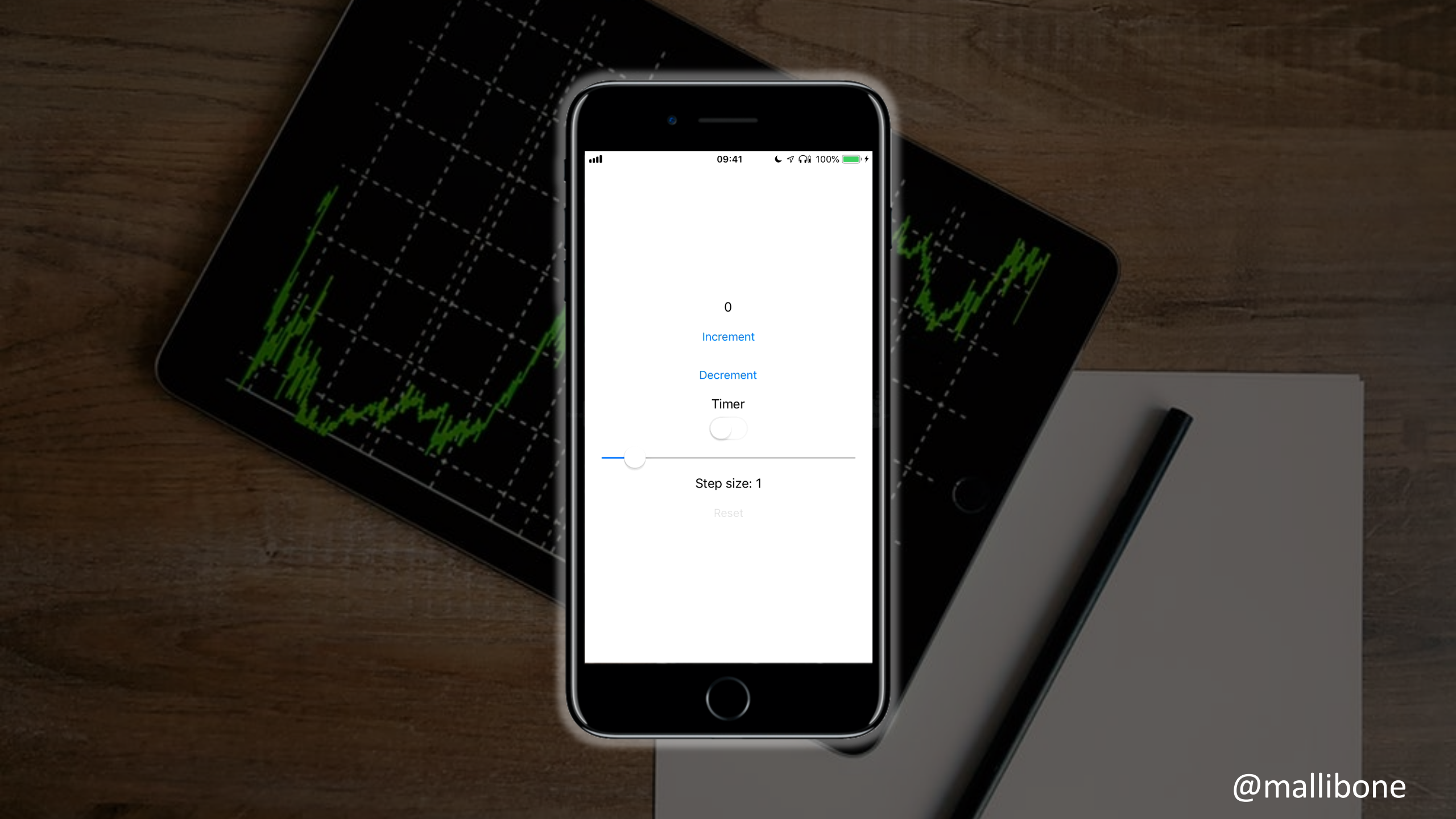
```
dotnet new -i Fabulous.XamarinForms.Templates
```

@mallibone

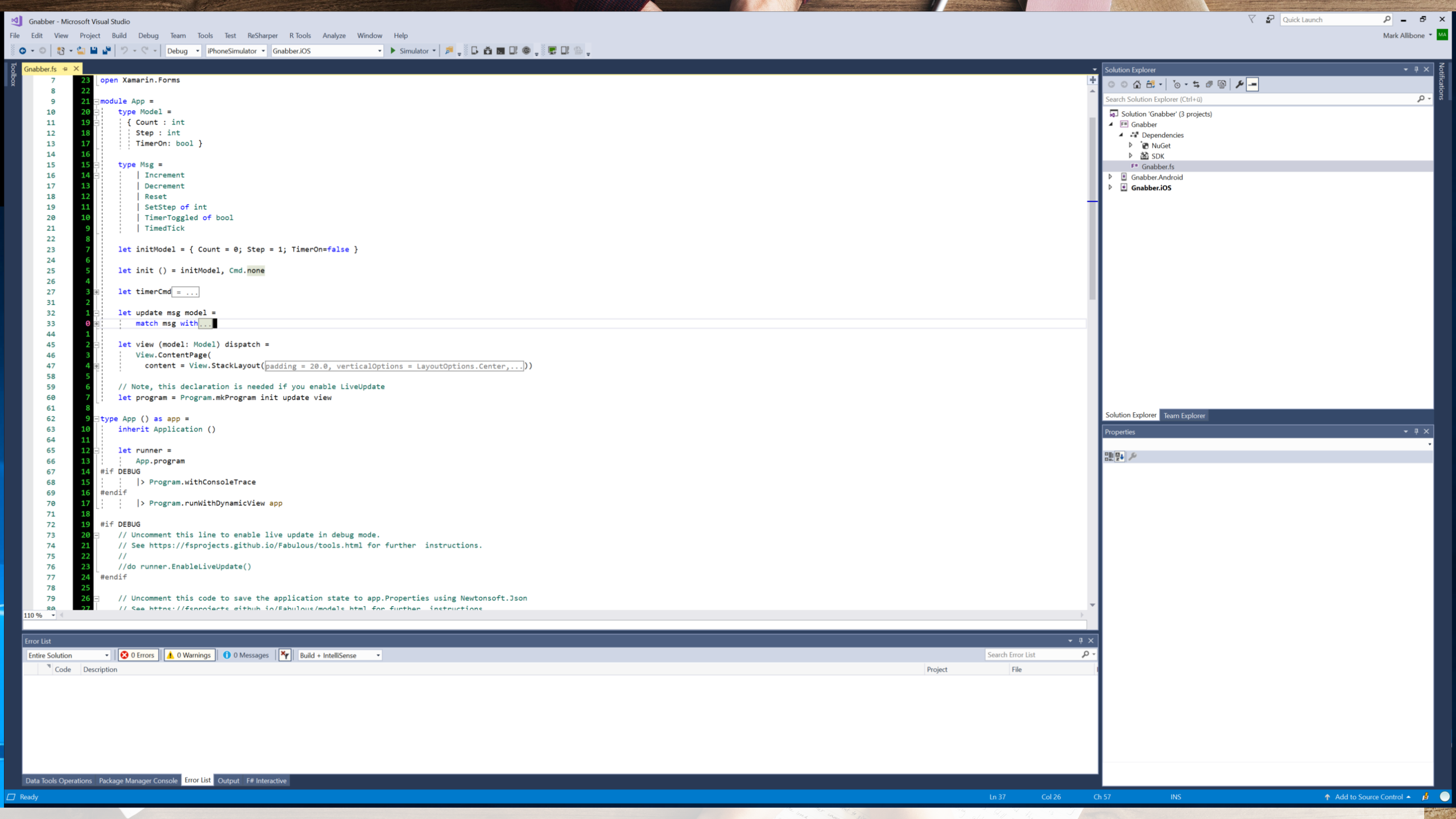

C:\Work\Playground
maal@CHREYNB0085 C:\Work\Playground [14:19]
dotnet new fabulous-xf-app -n Gnabber

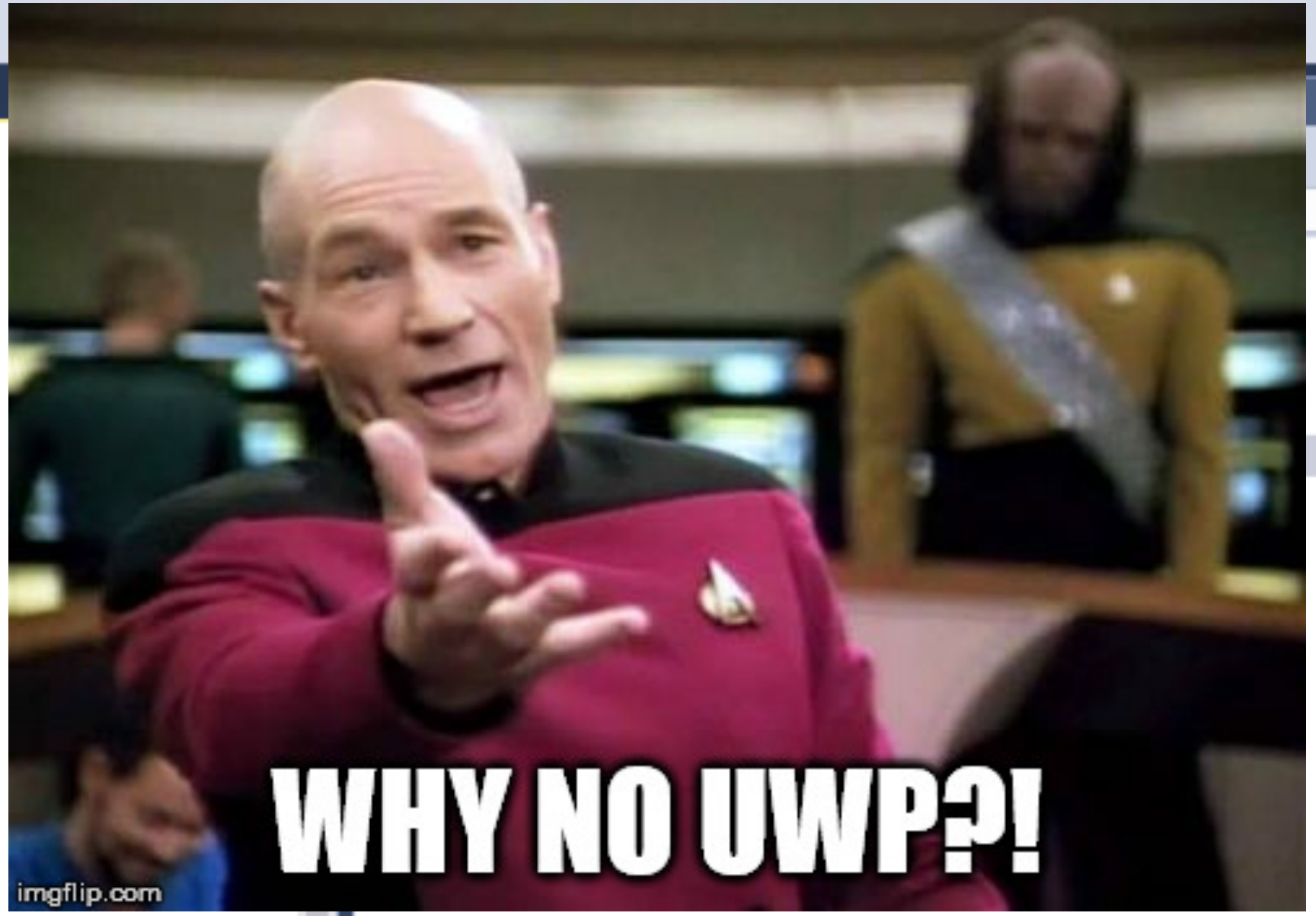
@mallibone

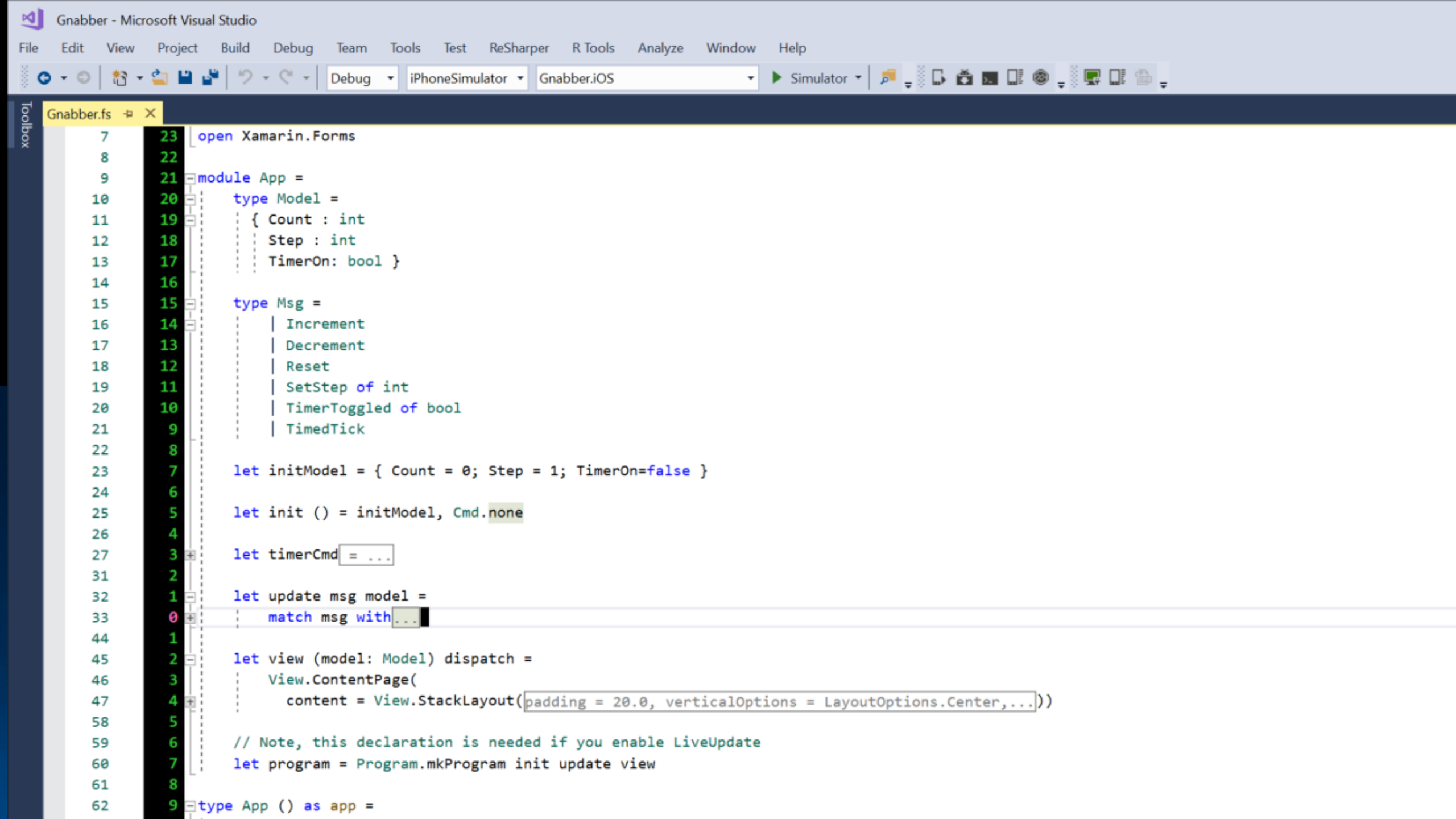

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Mark Allibone
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| Reset
| SetStep of int
| TimerToggled of bool
| TimedTick
let initModel = { Count = 0; Step = 1; TimerOn=false }
let init () = initModel, Cmd.none
let timerCmd = ...
let update msg model =
match msg with ...
let view (model: Model) dispatch =
View.ContentPage(
content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
// Note, this declaration is needed if you enable LiveUpdate
let program = Program.mkProgram init update view
type App () as app =
inherit Application ()
let runner =
App.program
#if DEBUG
|> Program.withConsoleTrace
#endif
|> Program.runWithDynamicView app
#if DEBUG
// Uncomment this line to enable live update in debug mode.
// See https://fsprojects.github.io/Fabulous/tools.html for further instructions.
//
//do runner.EnableLiveUpdate()
#endif
// Uncomment this code to save the application state to app.Properties using Newtonsoft.Json
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'Gnabber' (3 projects)
Gnabber
Dependencies
NuGet
SDK
Gnabber.fs
Gnabber.Android
Gnabber.iOS
Solution Explorer
Team Explorer
Properties
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Code
Description
Project
File
Data Tools Operations
Package Manager Console
Error List
Output
F# Interactive
Ready
Ln 37
Col 26
Ch 57
INS
Add to Source Control

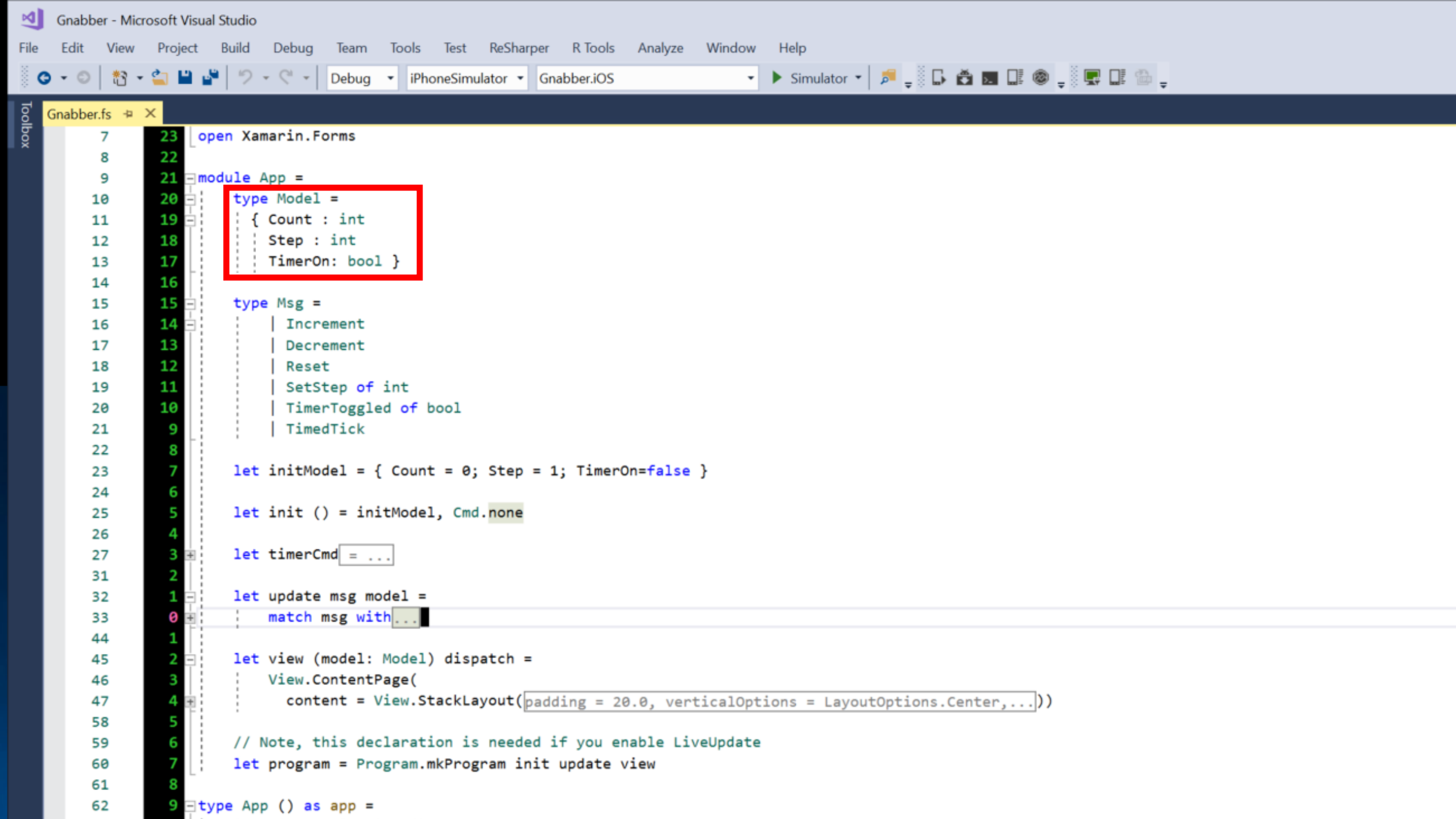


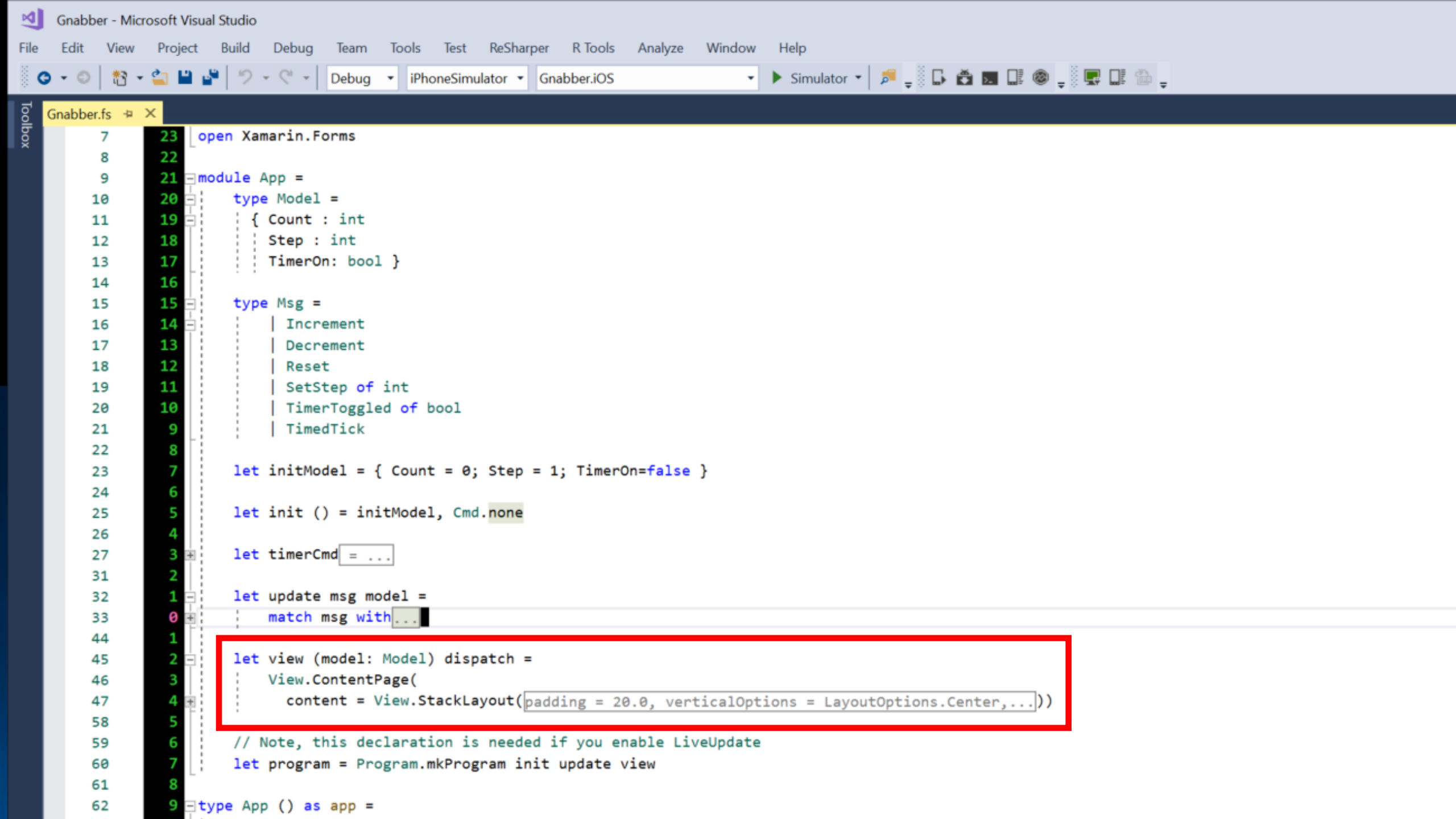
@mallibone

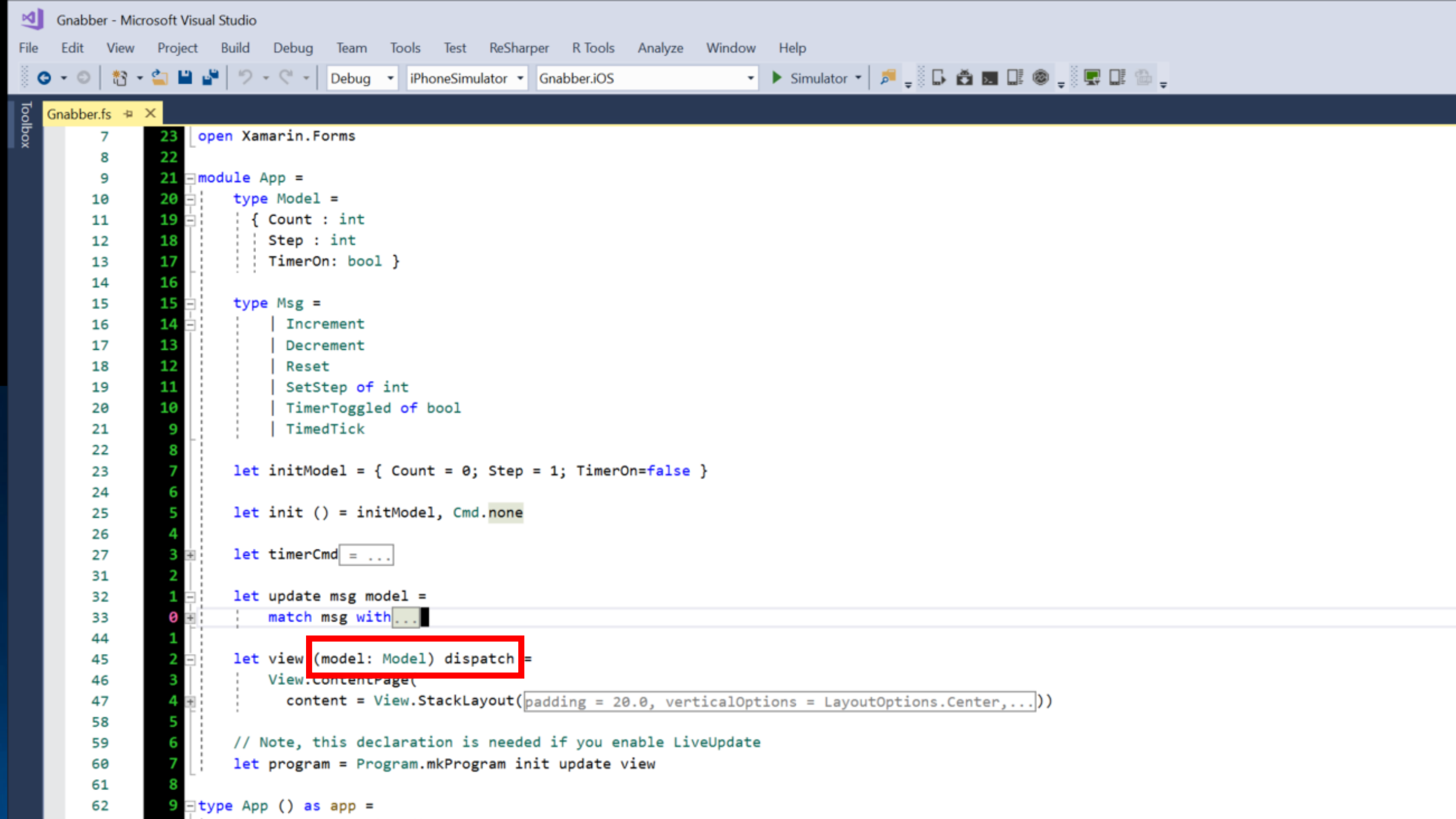

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Mark Allibone
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| Reset
| SetStep of int
| TimerToggled of bool
| TimedTick
let initModel = { Count = 0; Step = 1; TimerOn=false }
let init () = initModel, Cmd.none
let timerCmd = ...
let update msg model =
match msg with ...
let view (model: Model) dispatch =
View.ContentPage(
content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
// Note, this declaration is needed if you enable LiveUpdate
let program = Program.mkProgram init update view
type App () as app =
inherit Application ()
let runner =
App.program
#if DEBUG
|> Program.withConsoleTrace
#endif
|> Program.runWithDynamicView app
#if DEBUG
// Uncomment this line to enable live update in debug mode.
// See https://fsprojects.github.io/Fabulous/tools.html for further instructions.
//
//do runner.EnableLiveUpdate()
#endif
// Uncomment this code to save the application state to app.Properties using Newtonsoft.Json
Solution Explorer
Search Solution Explorer (Ctrl+ü)
Solution 'Gnabber' (3 projects)
Gnabber
Dependencies
NuGet
SDK
Gnabber.fs
Gnabber.Android
Gnabber.iOS
Solution Explorer
Team Explorer
Properties
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages
Build + IntelliSense
Search Error List
Code
Description
Project
File
Data Tools Operations
Package Manager Console
Error List
Output
F# Interactive
Ready
Ln 37
Col 26
Ch 57
INS
Add to Source Control

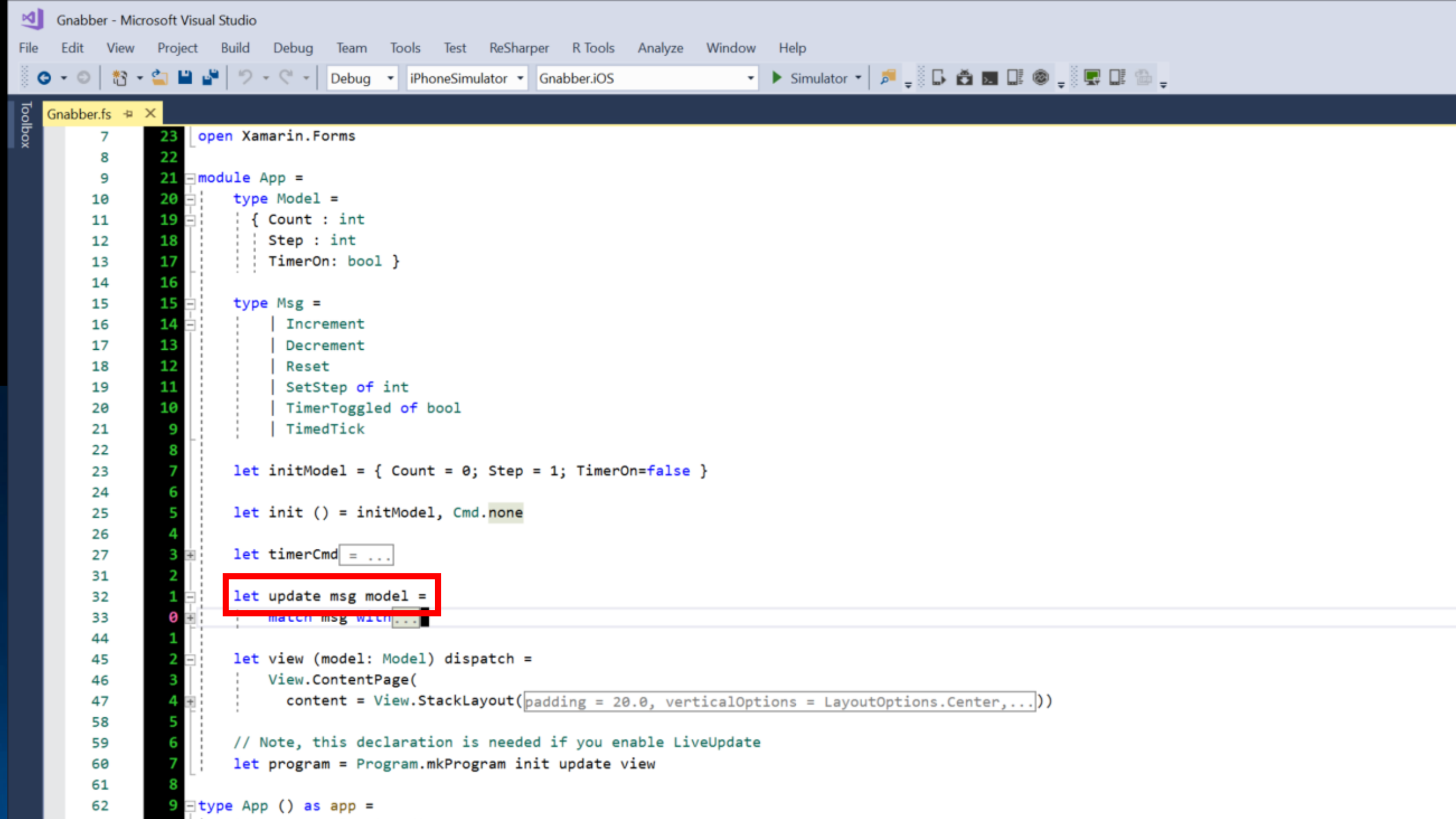

Quick Launch
Mark Allibone
MA
Notifications
WHY NO UWP?!
imgflip.com

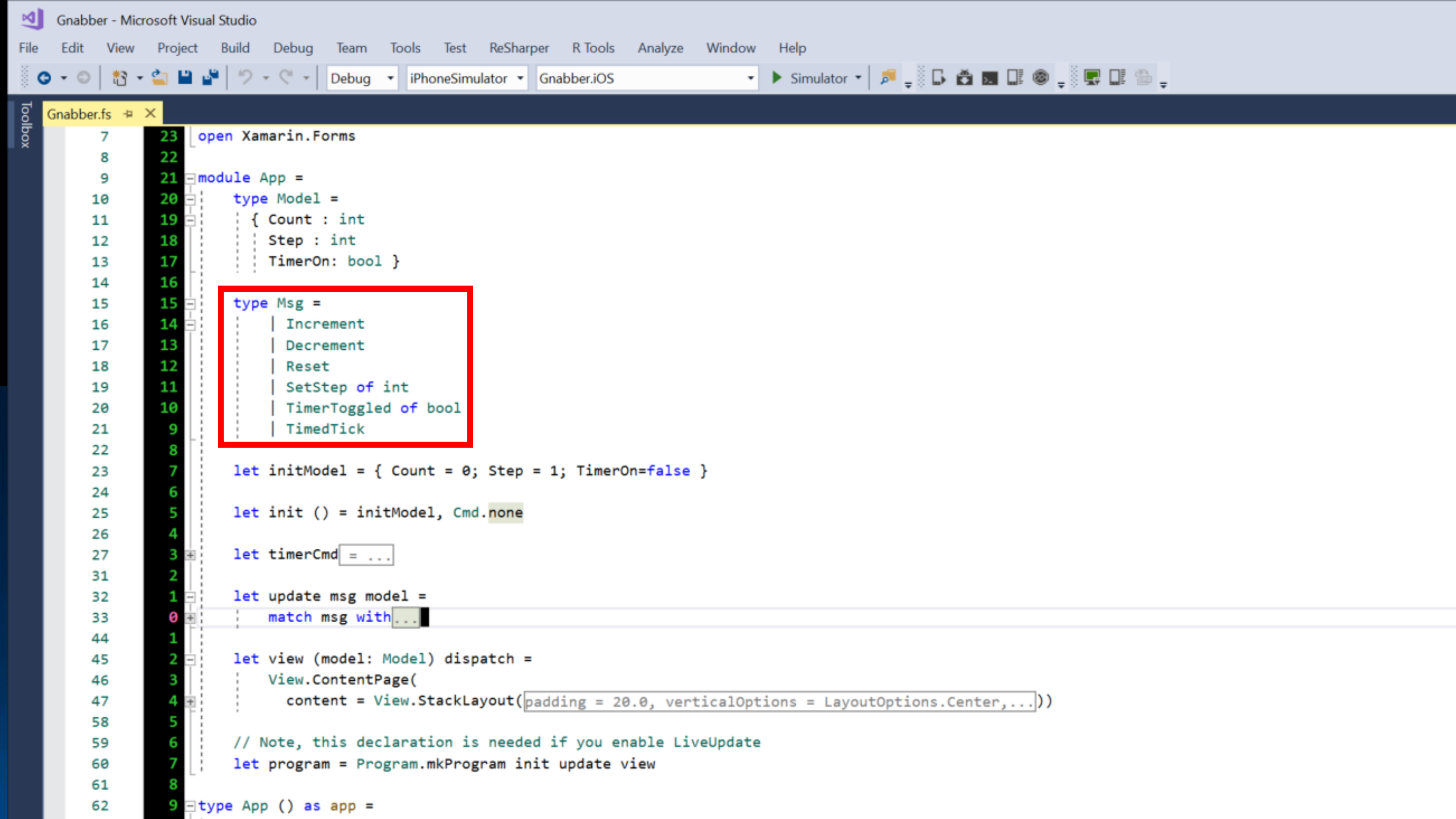

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| Reset
| SetStep of int
| TimerToggled of bool
| TimedTick
let initModel = { Count = 0; Step = 1; TimerOn=false }
let init () = initModel, Cmd.none
let timerCmd = ...
let update msg model =
match msg with...
let view (model: Model) dispatch =
View.ContentPage(
content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
// Note, this declaration is needed if you enable LiveUpdate
let program = Program.mkProgram init update view
type App () as app =


Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| Reset
| SetStep of int
| TimerToggled of bool
| TimedTick
let initModel = { Count = 0; Step = 1; TimerOn=false }
let init () = initModel, Cmd.none
let timerCmd = ...
let update msg model =
match msg with...
let view (model: Model) dispatch =
View.ContentPage(
content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
// Note, this declaration is needed if you enable LiveUpdate
let program = Program.mkProgram init update view
type App () as app =

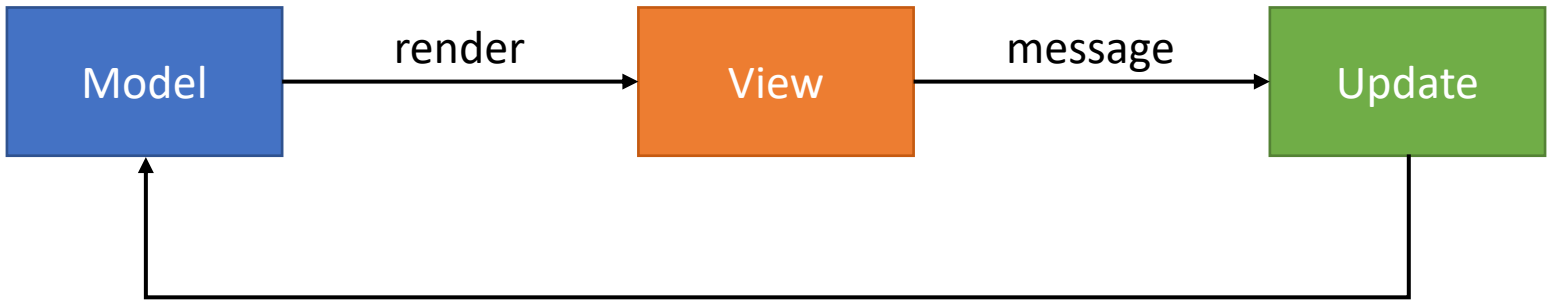

Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| Reset
| SetStep of int
| TimerToggled of bool
| TimedTick
let initModel = { Count = 0; Step = 1; TimerOn=false }
let init () = initModel, Cmd.none
let timerCmd = ...
let update msg model =
match msg with...
let view (model: Model) dispatch =
View.ContentPage(
content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
// Note, this declaration is needed if you enable LiveUpdate
let program = Program.mkProgram init update view
type App () as app =


Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| Reset
| SetStep of int
| TimerToggled of bool
| TimedTick
let initModel = { Count = 0; Step = 1; TimerOn=false }
let init () = initModel, Cmd.none
let timerCmd = ...
let update msg model =
match msg with...
let view (model: Model) dispatch =
View.ContentPage(
content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
// Note, this declaration is needed if you enable LiveUpdate
let program = Program.mkProgram init update view
type App () as app =


Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| Reset
| SetStep of int
| TimerToggled of bool
| TimedTick
let initModel = { Count = 0; Step = 1; TimerOn=false }
let init () = initModel, Cmd.none
let timerCmd = ...
let update msg model =
match msg with...
let view (model: Model) dispatch =
View.ContentPage(
content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
// Note, this declaration is needed if you enable LiveUpdate
let program = Program.mkProgram init update view
type App () as app =


Gnabber - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test ReSharper R Tools Analyze Window Help
Debug iPhoneSimulator Gnabber.iOS Simulator
Toolbox
Gnabber.fs
open Xamarin.Forms
module App =
type Model =
{ Count : int
Step : int
TimerOn: bool }
type Msg =
| Increment
| Decrement
| Reset
| SetStep of int
| TimerToggled of bool
| TimedTick
let initModel = { Count = 0; Step = 1; TimerOn=false }
let init () = initModel, Cmd.none
let timerCmd = ...
let update msg model =
match msg with...
let view (model: Model) dispatch =
View.ContentPage(
content = View.StackLayout(padding = 20.0, verticalOptions = LayoutOptions.Center,...))
// Note, this declaration is needed if you enable LiveUpdate
let program = Program.mkProgram init update view
type App () as app =

Model View Update




$\lim\left(1+\frac{1}{n}\right)^n = e$

- Driven from the model not the view
- State changes are defined with messages
- Single place for change - the update method
- Can be based on Xamarin Forms


Con un diseño como ese no habrá regalos para ti bajo el árbol de navidad

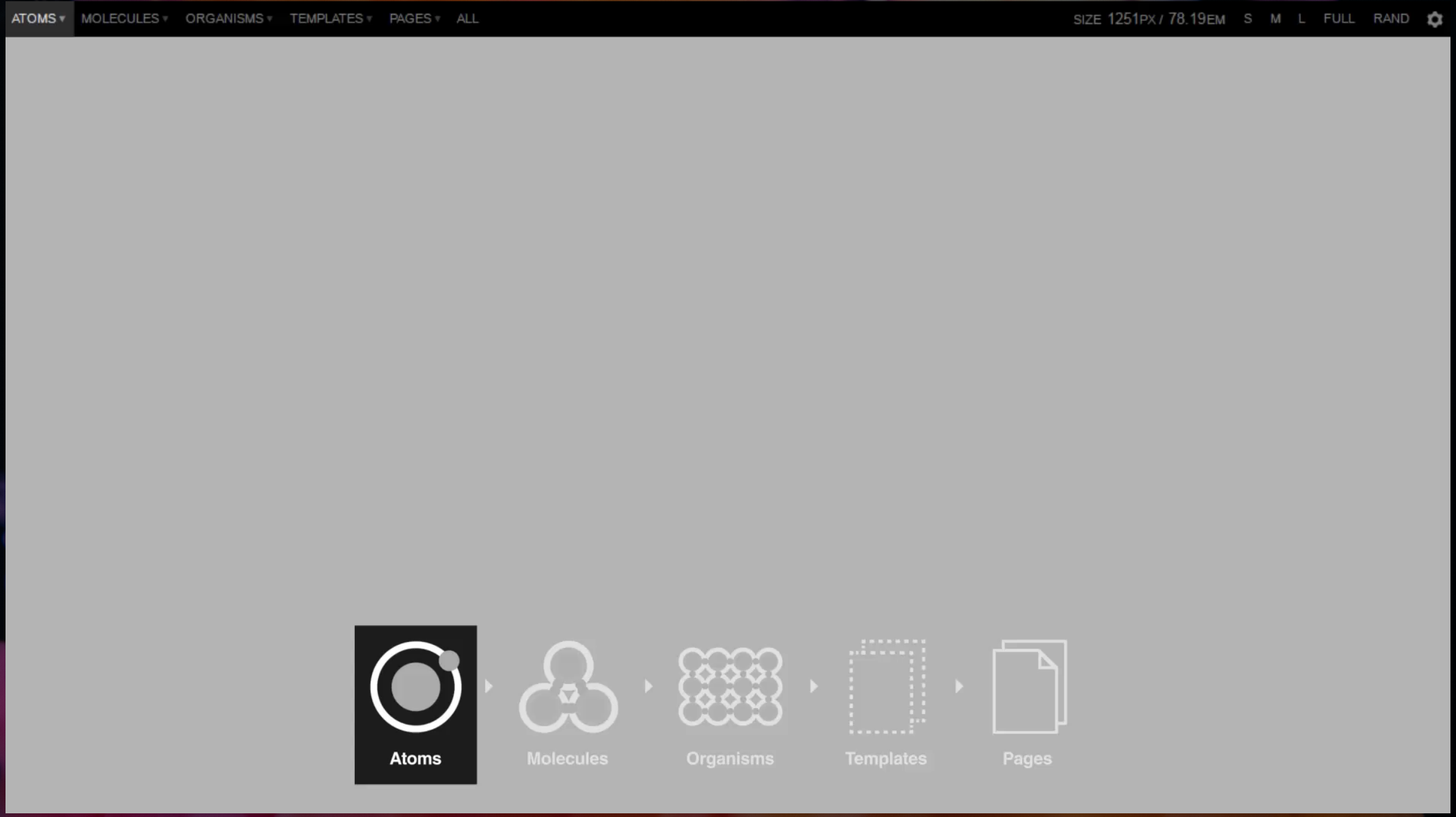
Doing it with style

@mallibone


XAML vs Coded UI


Demo
@mallibone

- Interactive Development
- Create reusable UI
- You can create good looking UIs
- UI is based on Xamarin Forms

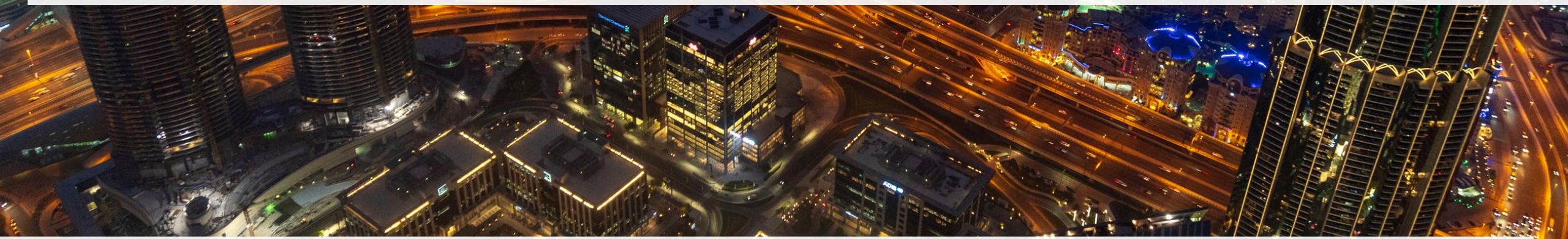


https://patternlab.io/

@mallibone


Live Update ❤️ Coded UI
https://fsprojects.github.io/Fabulous/tools.html
@mallibone

What about Shell? Visual Material? Or your other favorite Xamarin feature?



@mallibone

Xamarin.Essentials

04/22/2019 • 2 minutes to read • Contributors

Xamarin.Essentials provides developers with cross-platform APIs for their mobile applications.

Android, iOS, and UWP offer unique operating system and platform APIs that developers have access to all in C# leveraging Xamarin. Xamarin.Essentials provides a single cross-platform API that works with any Xamarin.Forms, Android, iOS, or UWP application that can be accessed from shared code no matter how the user interface is created.

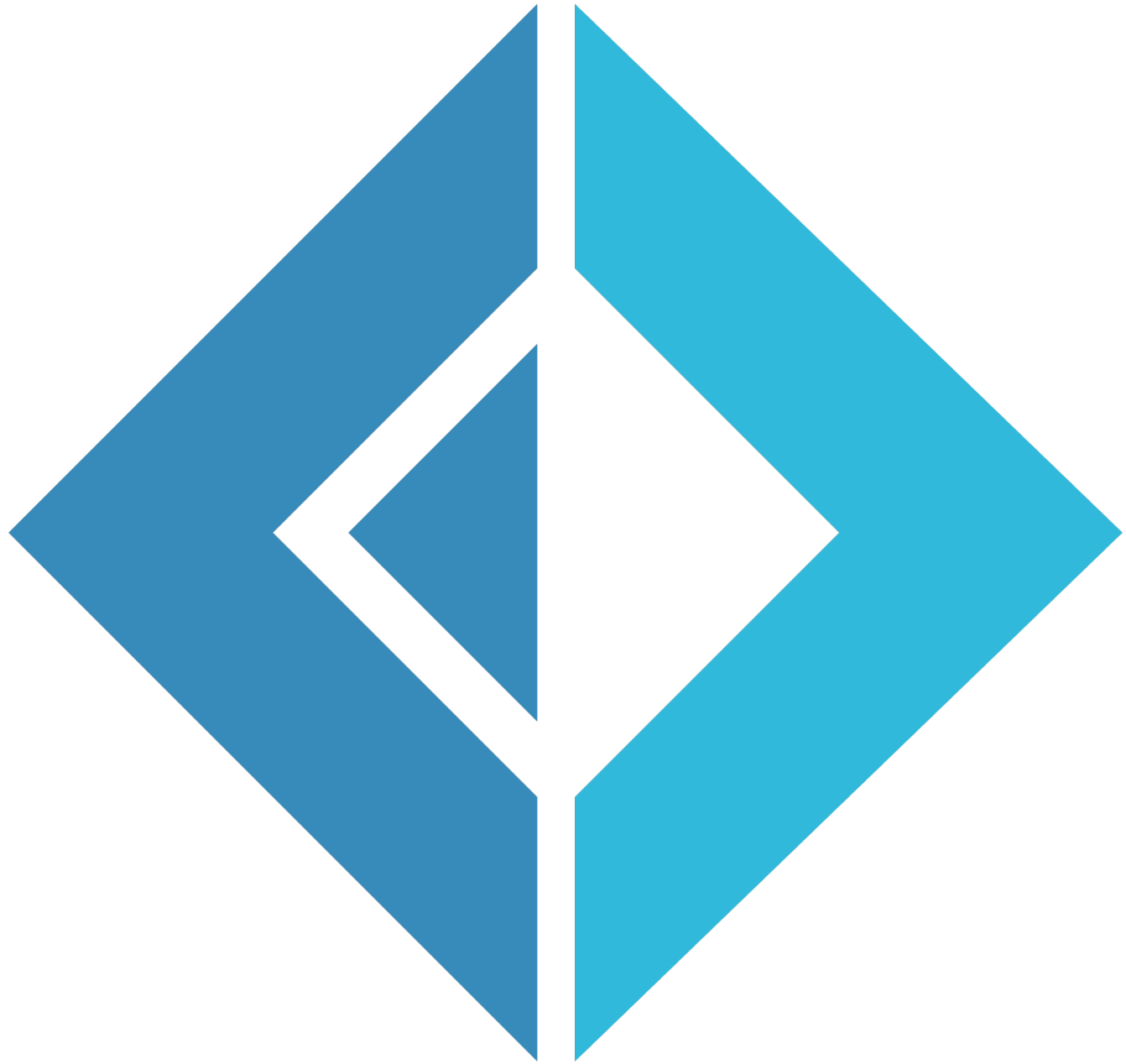
Get Started with Xamarin.Essentials

Follow the getting started guide to install the **Xamarin.Essentials** NuGet package into your existing or new Xamarin.Forms, Android, iOS, or UWP projects.

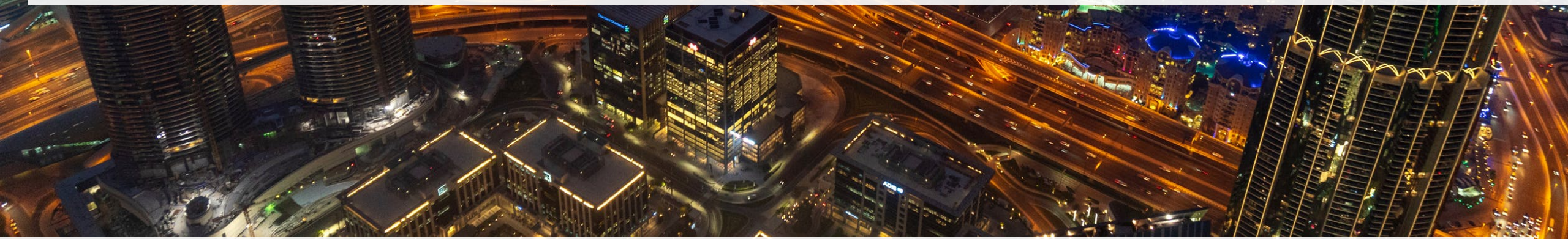
Feature Guides

Follow the guides to integrate these Xamarin.Essentials features into your applications:

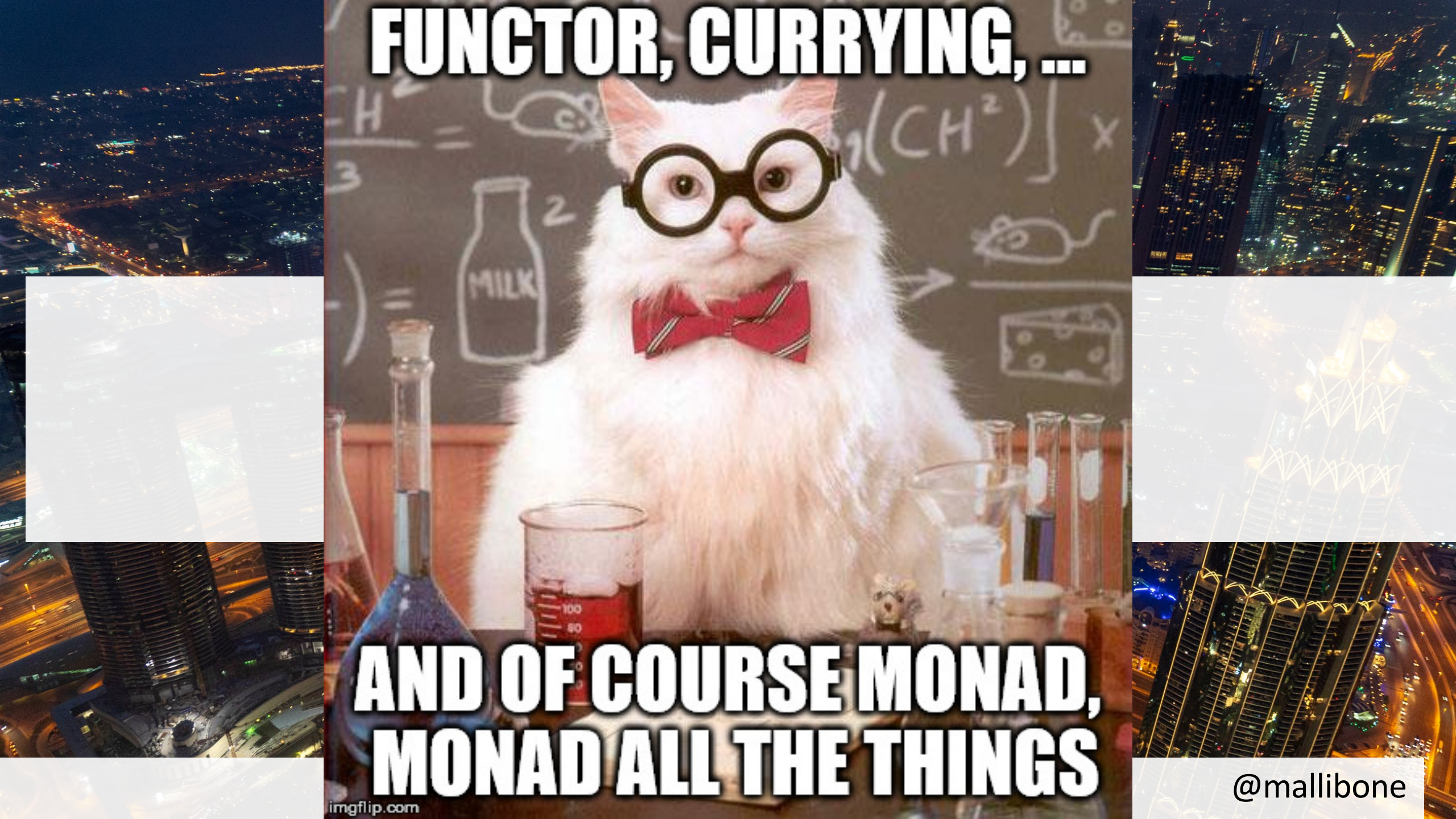
- Accelerometer – Retrieve acceleration data of the device in three dimensional space.
- App Information – Find out information about the application.
- Barometer – Monitor the barometer for pressure changes.
- Battery – Easily detect battery level, source, and state.
- Clipboard – Quickly and easily set or read text on the clipboard.
- Color Converters – Helper methods for System.Drawing.Color.
- Compass – Monitor compass for changes.
- Connectivity – Check connectivity state and detect changes.
- Detect Shake – Detect a shake movement of the device.
- Device Display Information – Get the device's screen metrics and orientation.
- Device Information – Find out about the device with ease.
- Email – Easily send email messages.
- File System Helpers – Easily save files to app data.
- Flashlight – A simple way to turn the flashlight on/off.
- Geocoding – Geocode and reverse geocode addresses and coordinates.
- Geolocation – Retrieve the device's GPS location.
- Gyroscope – Track rotation around the device's three primary axes.
- Launcher – Enables an application to open a URI by the system.
- Magnetometer – Detect device's orientation relative to Earth's magnetic field.
- MainThread – Run code on the application's main thread.
- Maps – Open the maps application to a specific location.
- Open Browser – Quickly and easily open a browser to a specific website.
- Orientation Sensor – Retrieve the orientation of the device in three dimensional space.
- Phone Dialer – Open the phone dialer.



How to get started with F#?



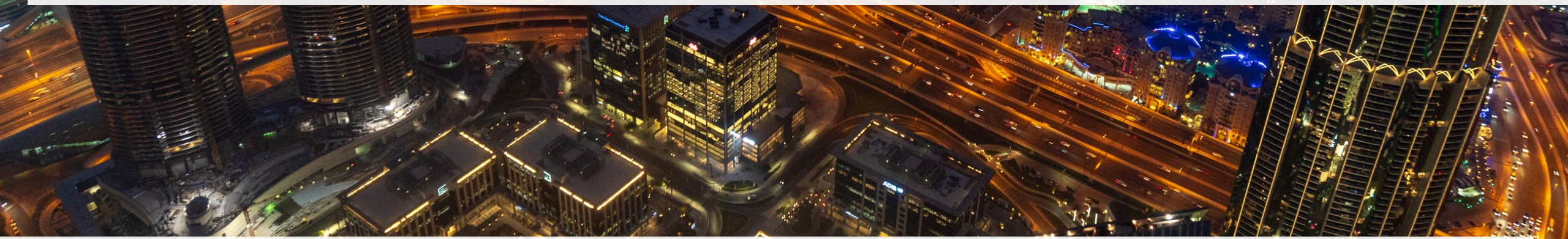
@mallibone


FUNCTOR, CURRYING, ...
AND OF COURSE MONAD,
MONAD ALL THE THINGS
imgflip.com

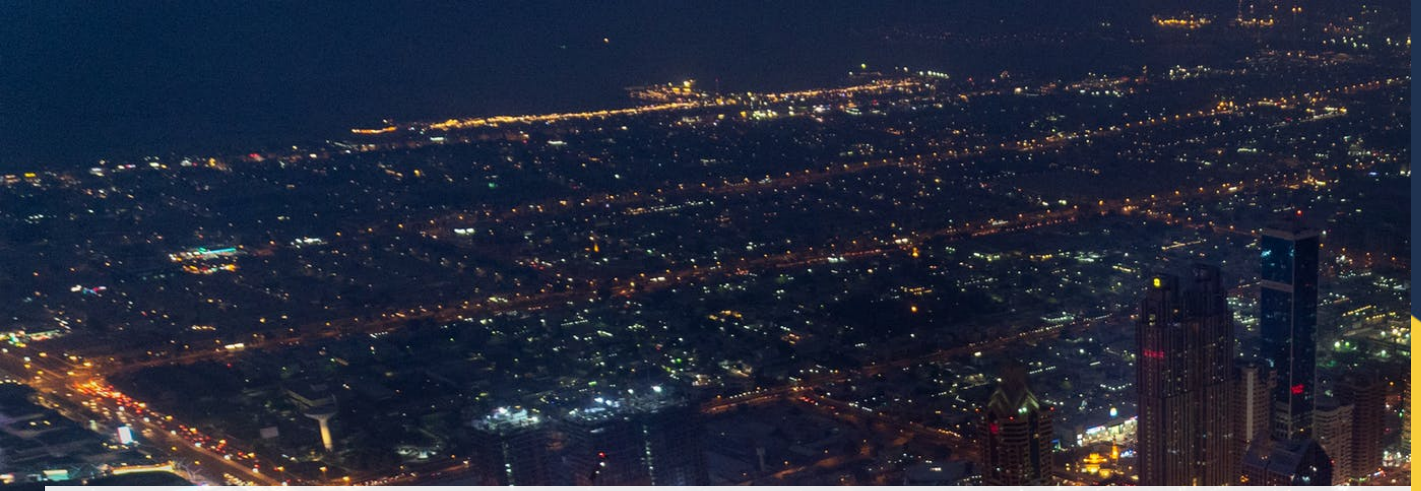
@mallibone


"HOLY FORKING SHIRT!"
#THEGOODPLACE
NBC

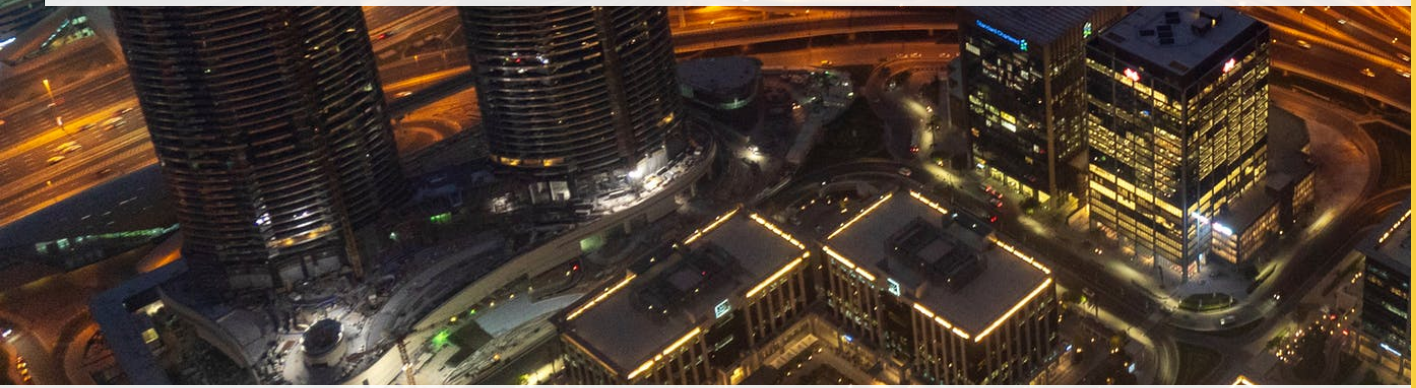
Learn Design patterns after learning the language

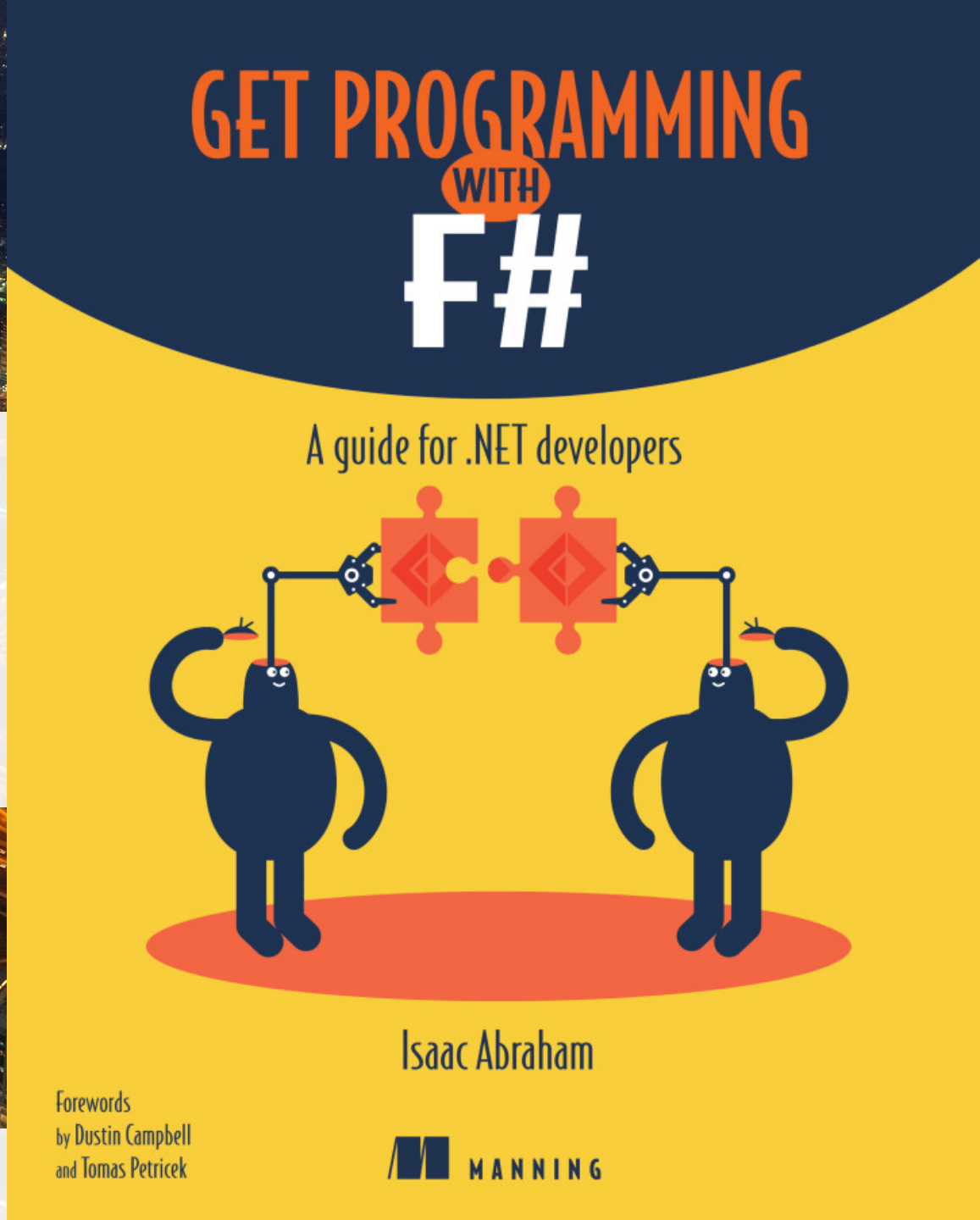


@mallibone



How to get started with F#?





REY TECHNOLOGY

Takeaways

REY TECHNOLOGY

Takeaways

- F# is not scary it's just your friendly functional first .NET language
- Model View Update simplifies state handling in UI
- Fabulous is everywhere
- Have a Fabulous Day!

REY TECHNOLOGY

Thank you for your time!

Mark Allibone @mallibone

Rey Technology

https://nullpointers.io

https://fsprojects.github.io/Fabulous

https://fsharpforfunandprofit.com/

FEATURED COMMUNITY BLOG https://mallibone.com

Microsoft® MVP Most Valuable Professional

#MonkeyConf

Gracias!

Gracias a nuestros Sponsors.
Sin ellos el evento no sería posible.

plain concepts

devs dna

Beezy

NAMOCODE

UXDIVERS
PRODUCT DESIGN STUDIO