Mobile and Desktop Applications with C♯

Rasmus Lystrøm
Associate Professor
ITU

Guest Lecture
Ali Reza Farahnak
Global Senior Customer Engineer
Mobile Application Development with C♯

Agenda

Mobile and Desktop Applications (by Ali)

- MVVM Architectural Pattern
- Xamarin.Forms
- .NET Multi-platform App UI (MAUI)

Upgrading .NET applications

C♯ 10

Catch up on other C♯ patterns

Blazor follow-up

Windows Applications

- Windows Forms (WinForms)
- Windows Presentation Foundation (WPF)

Upgrading .NET Applications

The Tools

```
dotnet tool install --global upgrade-assistant

dotnet tool install --global try-convert

upgrade-assistant analyze <csproj/sln>

upgrade-assistant upgrade <csproj/sln> --skip-backup
```

Update your stack

Get the latest .NET SDK: https://dot.net/

```
dotnet tool list --global
```

```
dotnet tool update --global dotnet-ef
```

```
dotnet list package --outdated
```

Update your .csproj files

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>
</Project>
```

Upgrading .NET Applications

Demo

C♯ 10 / .NET 6.0

Important new features

Global using directives

Implicit using directives

Implicit `Program.Main` from C♯ 9 is now default

File-scoped namespace declaration

Nullable reference types from C♯ 8 are now default

Hot reload

C♯ 10 / .NET 6.0

Demo

Catch up on other C♯ patterns

Parse vs. TryParse

[key] vs. TryGetValue

Tuple deconstruction

Discards

params

async Main

Reflection

Blazor follow-up

Demo

Mobile and Desktop Applications

UI Frameworks in .NET

Windows Forms (WinForms) (2002)

Windows Presentation Foundation (WPF) (2006)

Silverlight (2007-2019)

Xamarin.Forms (2014) (acquired by Microsoft in 2016)

Universal Windows Platform (UWP) (2015-2021?)

Windows UI Library (WinUI) (2021)

.NET Multi-platform App UI (MAUI) (2021)

XAML

XAML = eXtensible Application Markup Language

Windows Desktop (WPF)

Xamarin.Forms (iOS, Android, Windows)

.NET MAUI

WinUI

Silverlight (web)

Universal Windows Platform (anything windowsy)

XAML

Markup language for declaratively designing and creating application UIs

XAML maps XML markup to objects in the .NET Framework

Every tag maps to a class and every attribute to a property

Markup and procedural code are peers in functionality and performance

Code and markup are both first class citizens

Consistent model between UI, documents, and media

Compiled to code

XAML Markup vs. Code

```
<Button Width="100">OK
    <Button.Background>
        Purple
    </Button.Background>
</Button>
```


```
var button = new Button();
button.Content = "OK";
button.Background = new SolidColorBrush(Colors.Purple);
button.Width = 100;
```

MainPage.xaml

```
<Page>
    <Grid>
        <StackPanel>
            <Ellipse Name="Light" Fill="Red"
                     Height="200" Width="200" Margin="50" />
            <Button Width="150"
                    Content="Change Lights"
                    HorizontalAlignment="Center"
                    Click="Button_Click" />
        </StackPanel>
    </Grid>
</Page>
```

MainPage.xaml.cs

```
namespace App
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        private void Button_Click(object sender, RoutedEventArgs e)
        {
            var current = Light.Fill as SolidColorBrush;

            if (current.Color == Colors.Red)
                Light.Fill = new SolidColorBrush(Colors.Green);
            else
                Light.Fill = new SolidColorBrush(Colors.Red);
        }
    }
}
```

Desktop Applications

Demo

Image source: http://lazergaze.tumblr.com/post/26333564955

MVVM

The Model-View-ViewModel Pattern

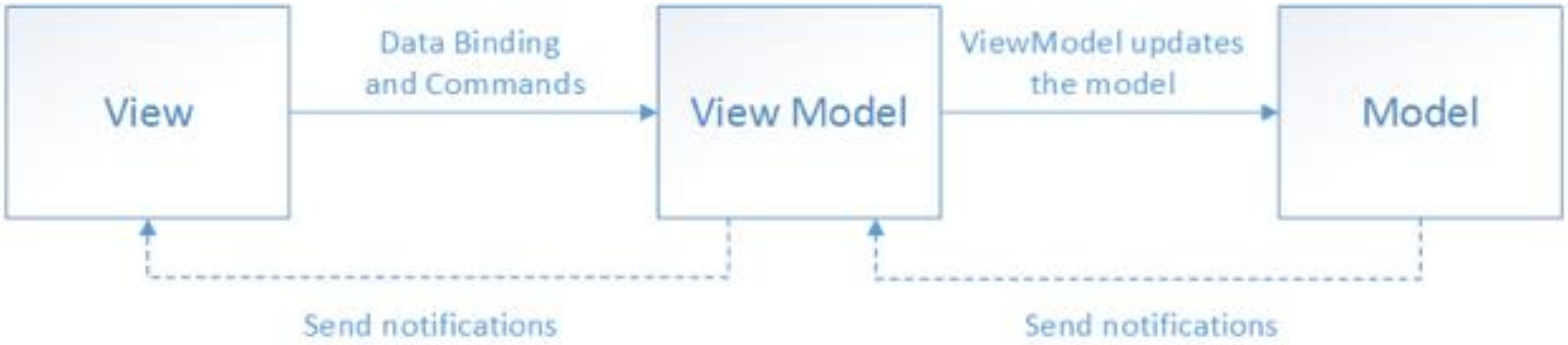
Separation of logic and presentation

Having event handlers in the code-behind is bad for testing, since you cannot mock away the view

Changing the design of the view often also requires changes in the code, since every element has its different event handlers

The logic is tightly bound to the view. It's not possible to reuse the logic in an other view

MVVM


View
Data Binding
and Commands
View Model
ViewModel updates
the model
Model
Send notifications
Send notifications

MVVM

Demo

MVVM concepts

There is conceptually only ever one MODEL

Code in code-behind should be ABSOLUTELY MINIMAL

A ViewModel should ALWAYS implement `INotifyPropertyChanged`

A ViewModel may be used for more than one view

MVVM Design Patterns

Observer Pattern:

- `INotifyPropertyChanged`
- `ObservableCollection<T>`

Command Pattern:

- `ICommand`

Thank You