

```

// R_FatNumForName (SKY1NAME)
// If no world levels, no secret exit
// terminates the sky texture to be used
// on the current episode, and the game version.
// code == commercial
// gamemode == pack (tr)
// gamemode == pack (plut 1)
// secrettext = false;
// secrettext = true;
// gamemode == ga (completed);
// gamemode == ga (nothing);

// skip the description field
// memmap (check, 0, sizeof(check))
// sprintf (check, "version %d", VERSION);
// if (strcmp (base.p, check))
// return; // bad version
// save_p += VERSIONSIZE;

// This was quite messy with SPECIAL and commented parts.
// supposedly backs to make the latest edition work.
// it might not work properly.
// (episode < 1)
// episode = 1;

// (gamemode == retail)
// (episode < 1)
// episode = 4;
// else if (gamemode == shareware)
// (episode < 3)
// episode = 1; // only start episode 1 on shareware
// else
// (episode < 3)
// episode = 3;

// (map < 1)
// map = 1;
// if (map > 90)
// (gamemode != commercial)
// map = 9;
// M_ClearRandom (0);

// (skill == sk_nightmare || respawnparm)
// respawnmonsters = true;
// else
// respawnmonsters = false;
// (lastparm || (skill == sk_nightmare && gameskill != sk_nightmare))
// for (i=S_SARG_RUN1; i<=S_SARG_RUN2; i++)
// statesh.tics >= 1;
// mobinfo[MT_BRUISERSHOT].speed = 20*FRACUNIT;
// mobinfo[MT_TROOPSHOT].speed = 20*FRACUNIT;
// else if (skill != sk_nightmare && gameskill != sk_nightmare)
// for (i=S_SARG_RUN1; i<=S_SARG_RUN2; i++)
// statesh.tics <= 1;
// mobinfo[MT_BRUISERSHOT].speed = 15*FRACUNIT;
// mobinfo[MT_HEADSHOT].speed = 10*FRACUNIT;
// mobinfo[MT_TROOPSHOT].speed = 10*FRACUNIT;

// force players to be initialized upon first level load
// for (i=0; i<MAXPLAYERS; i++)
// players[i].playerstate = PST_REBORN;

// usergame = true; // will be set, false if a demo
// paused = false;
// demoplayback = false;
// automapactive = false;
// viewactive = true;
// gamemode = episode;
// gamemapsof = map;
// gameskill = skill;
// viewactive = true;

```

UI Programming in .NET XAML Crash Course

Rasmus Lystrøm
External Associate Professor
ITU

Agenda

MVC / WebAPI recap

XAML and the Universal Windows Platform

Disclaimer

I'm not a UI expert...



XAML = eXtensible Application Markup Language

Windows Desktop (WPF)

Windows Universal (anything)

Xamarin Forms (iOS, Android, Windows)

Silverlight (web)



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

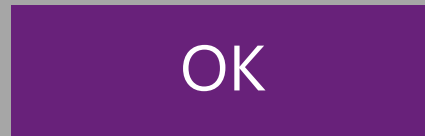
Code and markup are both first class citizens

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
  x:Class="App.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">
  <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);
        }
    }
}
```


XAML DEMO

XAML + code-behind



```
{  
    Light.Fill = new SolidColorBrush(Colors.Red);  
}  
}
```

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

The Model-View-ViewModel Pattern

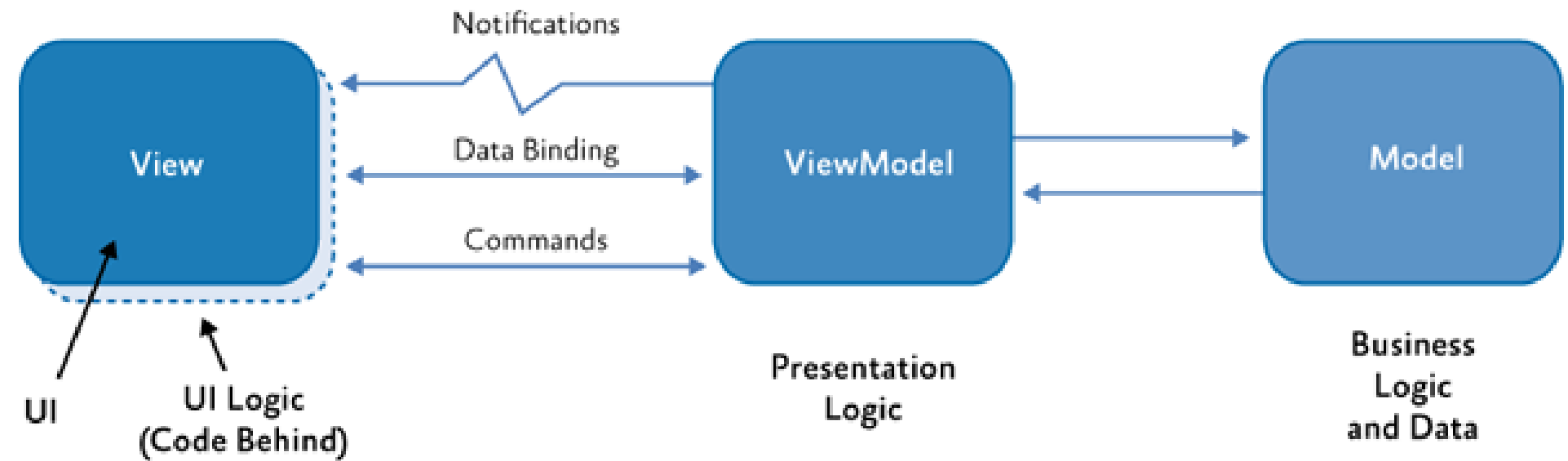
Separation of logic and presentation

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 it's different event handlers

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

MVVM



MVVM DEMO

MVVM: Pieces of the puzzle

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: Tips

Do

MV

Te



**Get your hands
dirty first!**