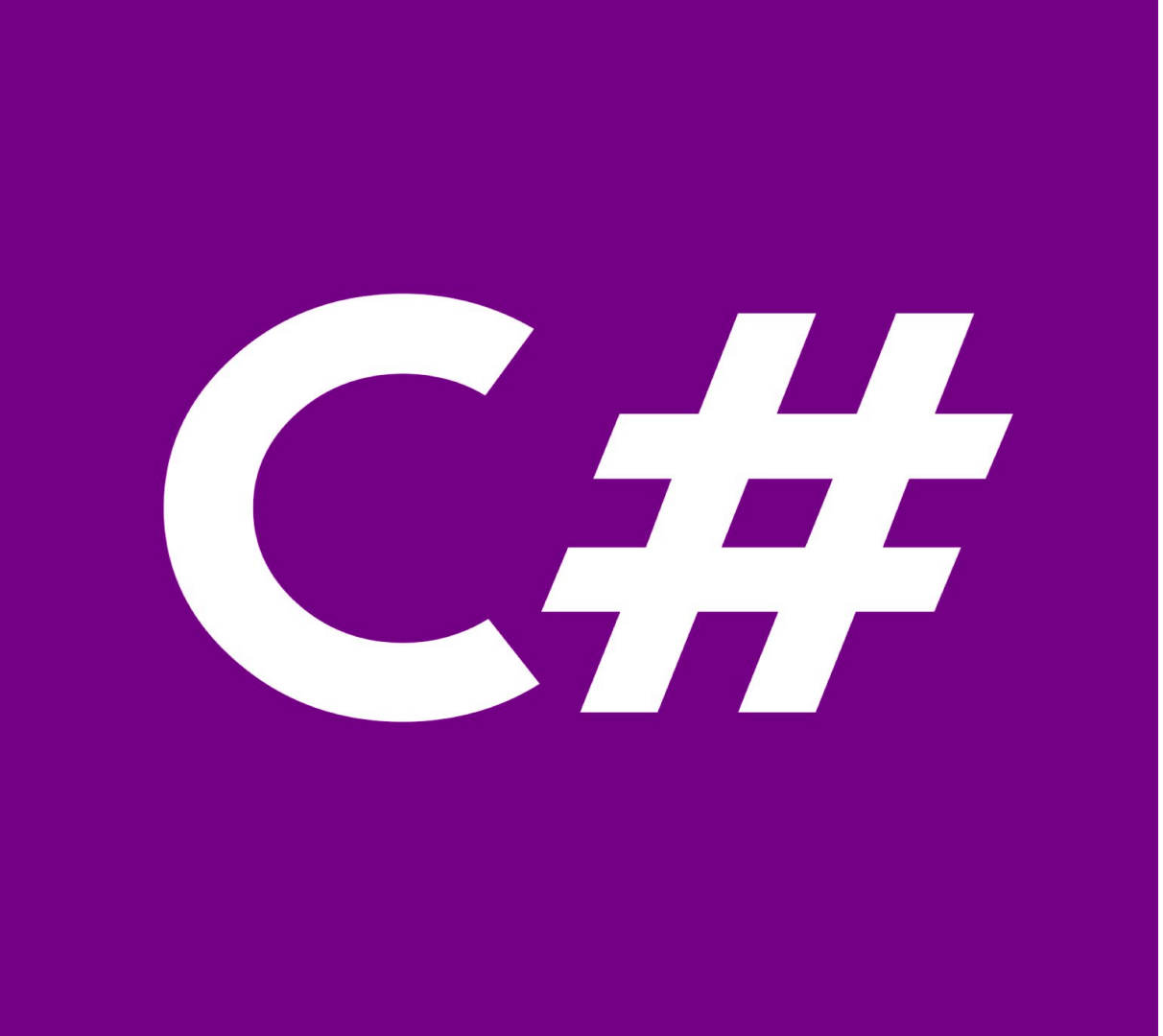
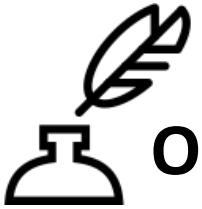


TDDD49
C# and .NET Programming

(Lecture 03)

Sahand Sadjadee
Department of Information and Computer Science
Linköping University



Outline

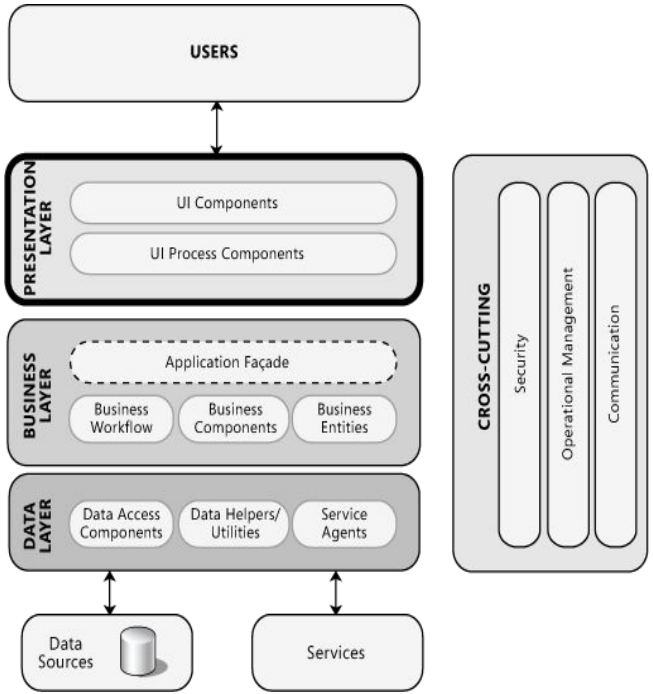
1. The Presentation Layer
2. **W**indows **P**resentation **F**oundation (WPF)
3. MVVM Design Pattern

LINKÖPINGS UNIVERSITET

The Presentation Layer

The Presentation Layer

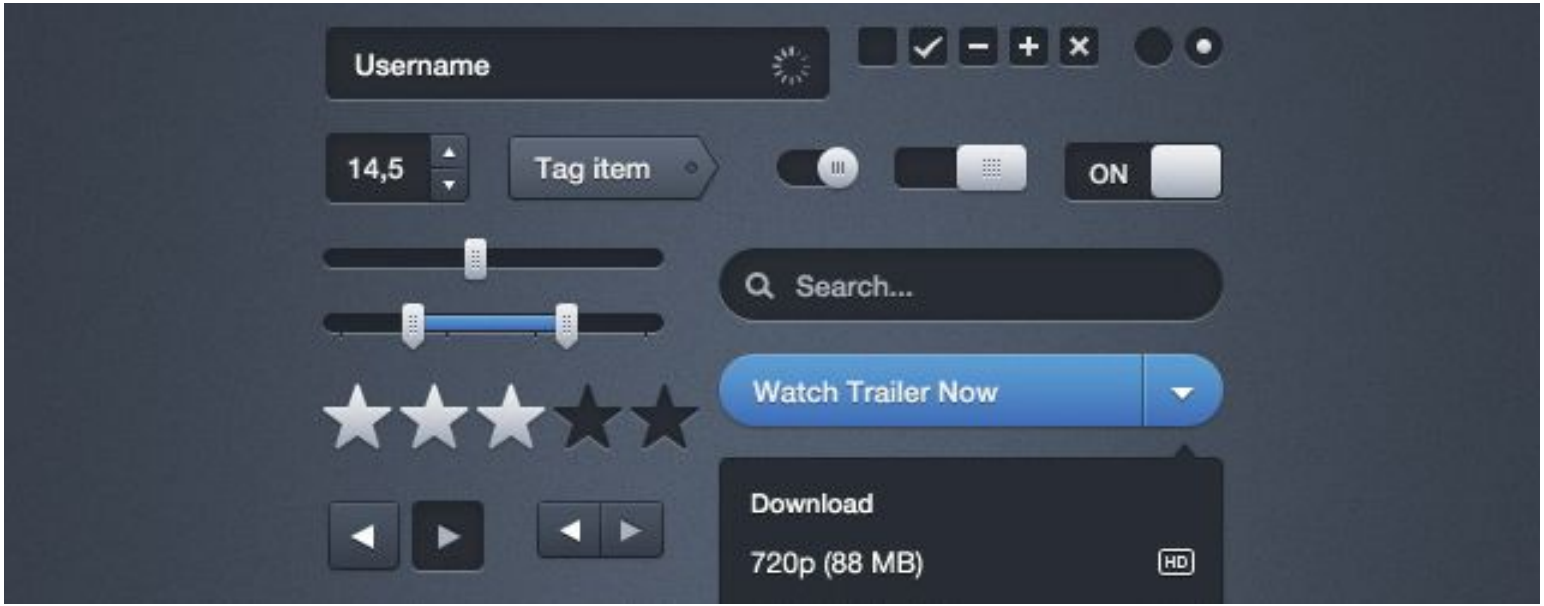
https://msdn.microsoft.com/en-us/library/ff647339.aspx#diforwc-ch01_definingthepresentationlayer



- **User Interface components**. These are the application's visual elements used to display information to the user and accept user input.
- **Presentation Logic components**. Presentation logic is the application code that defines the logical behavior and structure of the application in a way that is independent of any specific user interface implementation. When implementing the Separated Presentation pattern, the presentation logic components may include Presenter, Presentation Model, and ViewModel components. The presentation layer may also include Presentation Layer Model components that encapsulate the data from your business layer, or Presentation Entity components that encapsulate business logic and data in a form that is easily consumable by the presentation layer.

LINKÖPINGS UNIVERSITET

User Interface Elements/Components



LINKÖPINGS UNIVERSITET

The Presentation Layer

https://msdn.microsoft.com/en-us/library/ff647339.aspx#diforwc-ch01_definingthepresentationlayer

- **User interface components**
  - Acquiring data from the user
  - Rendering data to the user
  - Validation, input masking, and using appropriate controls for data input
  - Managing visual layouts, styles, and the general appearance and navigation of the application
  - Encapsulating the effect of globalization and localization
  - Formatting data and displaying it in useful visual styles
  - Browsing, searching, and organizing displayed data

The Presentation Layer Design Considerattions https://msdn.microsoft.com/en-us/library/ee658081.aspx

- Choose the appropriate application type.
- Choose the appropriate UI technology.
- **Use the relevant patterns.**
- Design for separation of concerns.
- Consider human interface guidelines.
- Adhere to user driven design principles.

LINKÖPINGS UNIVERSITET

The Presentation Layer Design Issues

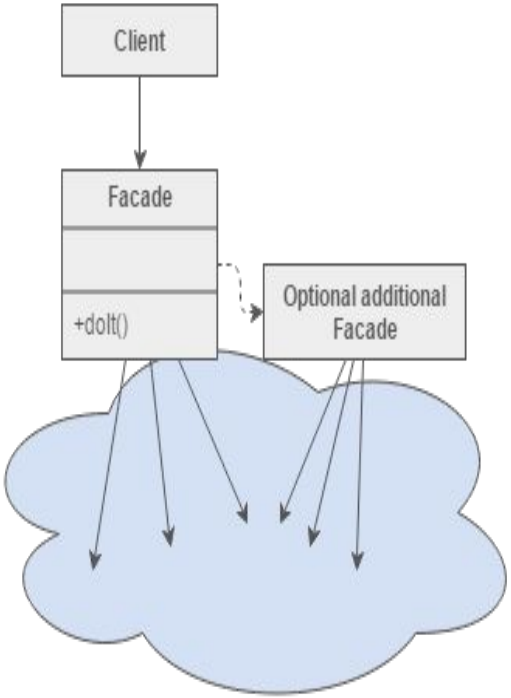
https://msdn.microsoft.com/en-us/library/ee658081.aspx

- Caching
- Communication
- Composition
- Exception Management
- Navigation
- User Experience
- User Interface
- Validation

Category	Relevant patterns
Caching	**Cache Dependency.** Use external information to determine the state of data stored in a cache. **Page Cache.** Improve the response time for dynamic Web pages that are accessed frequently, but change less often and consume a large amount of system resources to construct.
Composition and Layout	**Composite View**. Combine individual views into a composite representation. **Presentation Model** (Model-View-ViewModel) pattern. A variation of Model-View-Controller (MVC) tailored for modern UI development platforms where the View is the responsibility of a designer rather than a classic developer. **Template View**. Implement a common template view, and derive or construct views using this template view.
Exception Management	**Exception Shielding**. Prevent a service from exposing information about its internal implementation when an exception occurs.
Navigation	**Application Controller**. A single point for handling screen navigation.
User Experience	**Asynchronous Callback**. Execute long-running tasks on a separate thread that executes in the background, and provide a function for the thread to call back into when the task is complete. **Chain of Responsibility**. Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request.

Facade Design Pattern

https://sourcemaking.com/design_patterns/facade



Intent

- Provide a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use.
- Wrap a complicated subsystem with a simpler interface.

Problem

A segment of the client community needs a simplified interface to the overall functionality of a complex subsystem.

Can be used in the Logic Layer? How it helps?

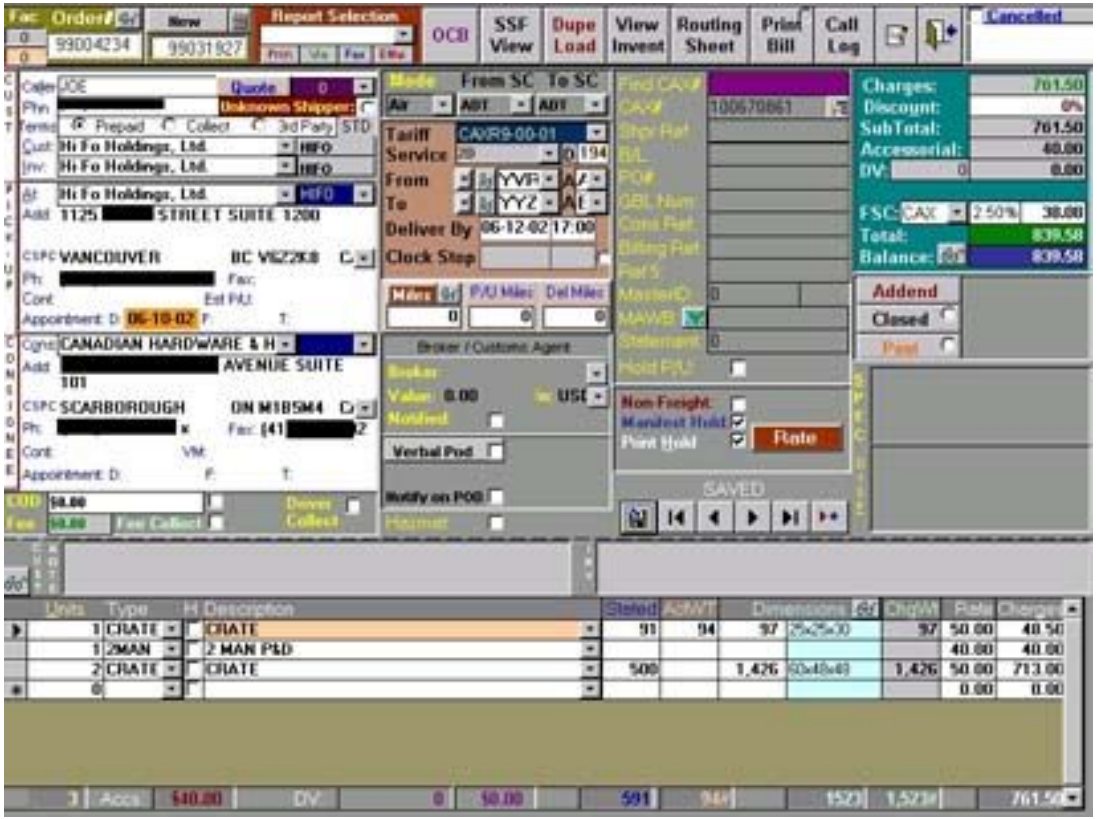
LINKÖPINGS UNIVERSITET

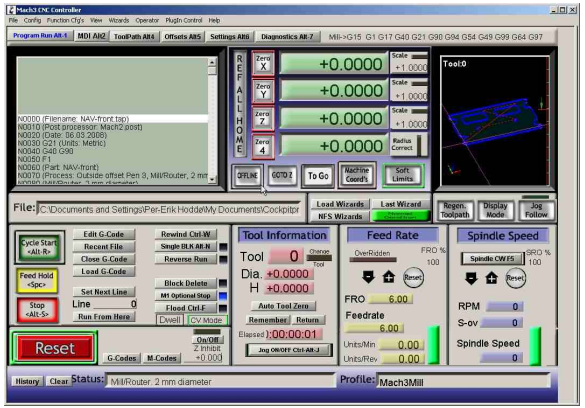
The presentation layer contains the UI components and more…

User Interface Design Principles

- Clarity is job #1
- Interfaces exist to enable interaction
- One primary action per screen
- Provide a natural next step
- Consistency matters
- Strong visual hierarchies work best
- A crucial moment: the zero state
- And more… http://bokardo.com/principles-of-user-interface-design/

User Interface - bad design examples...





LINKÖPINGS UNIVERSITET

Windows Presentation Foundation

WPF

https://msdn.microsoft.com/en-us/library/aa970268(v=vs.100).aspx

- Windows Presentation Foundation (WPF) is a next-generation presentation system for building Windows client applications, desktop applications, with visually stunning user experiences.
- With WPF, you can create a wide range of both standalone and browser-hosted applications.
- The WPF development platform supports a broad set of application development features, including an application model, resources, controls, graphics, layout, data binding, documents, and security.
- WPF is a standard part of .NET framework since version 3.0.

WPF application types

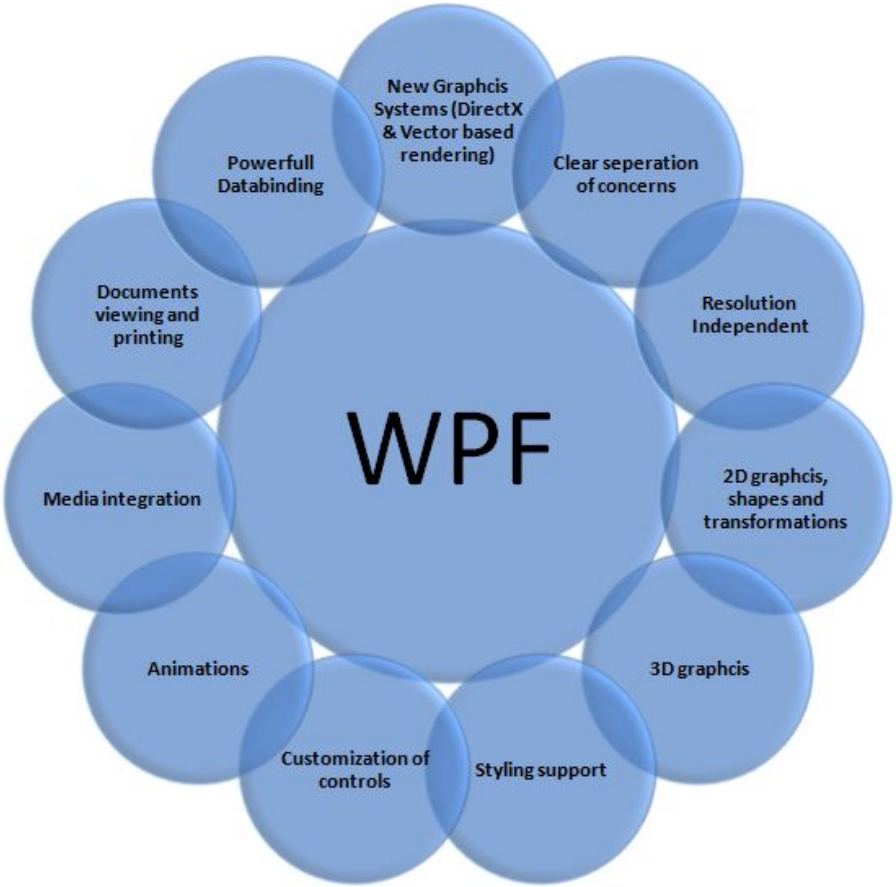
http://paxcel.net/blog/opting-for-right-wpf-application-type/

There are three different types of WPF Applications

- **Traditional Desktop Applications**
- Navigation Based WPF Application
- WPF Browser Hosted Applications (XBAP)

`System.Windows` Namespaces

https://msdn.microsoft.com/en-us/library/gg145013(v=vs.110).aspx



`System.Windows` contains all the namespaces and classes which form WPF framework.

https://msdn.microsoft.com/en-us/library/aa970268%28v=vs.100%29.aspx

LINKÖPINGS UNIVERSITET

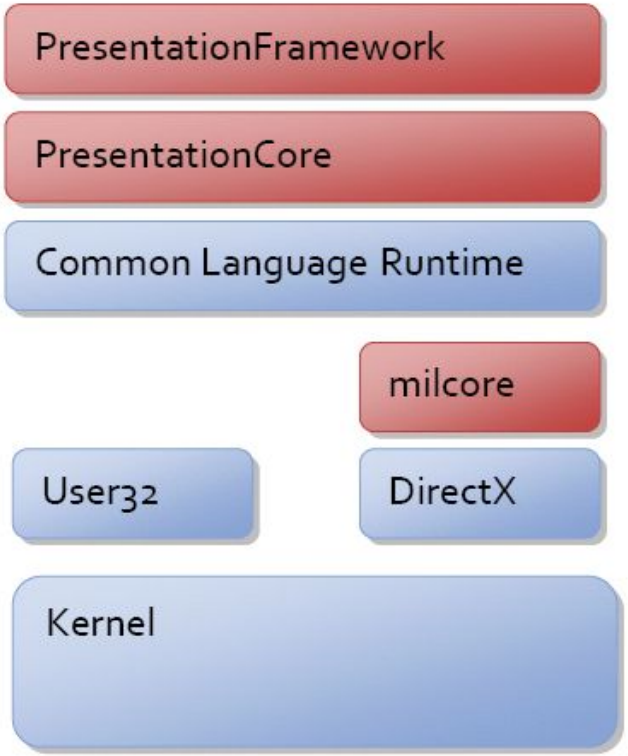
Standard WPF Controls

https://msdn.microsoft.com/en-us/library/bb655881(v=vs.90).aspx

Control name	Description
System.Windows.Controls.Border	Displays a border around content.
System.Windows.Controls.Button	Enables a user to perform an action by clicking a button. The `Buttonbase.Click` event occurs when a**Button** is clicked.
System.Windows.Controls.CheckBox	Enables a user to select and clear a check box to indicate a Yes/No or True/False value.
System.Windows.Controls.ComboBox	Enables a user to select an item from a drop-down list. The list is displayed when the user clicks a drop-down arrow.
System.Windows.Controls.Grid	Defines a flexible grid area that consists of columns and rows.
System.Windows.Controls.Image	Displays an image.
System.Windows.Controls.Label	Displays text on a form. Provides support for access keys.
System.Windows.Controls.ListBox	Enables a user to select an item from a list.
System.Windows.Controls.RadioButton	Enables a user to choose from among mutually exclusive items. The selection of one radio button is mutually exclusive to any other radio button in the same container.
System.Windows.Controls.StackPanel	Enables you to stack child controls vertically or horizontally.
System.Windows.Control.TabControl	Enables visual content to be arranged in a tabular form.
System.Windows.Controls.TextBox	Displays unformatted text and enables users to enter text.

WPF Architecture

https://msdn.microsoft.com/en-us/library/ms750441(v=vs.110).aspx



Key classes:

System.Threading.DispatcherObject

System.Windows.DependencyObject

System.Windows.Media.Visual

System.Windows.UIElement

System.Windows.FrameworkElement

System.Windows.Controls.Control

LINKÖPINGS UNIVERSITET

WPF - sample code (C#)

WPF https://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx

```
using System.Windows;
using System.Windows.Controls;
using System.Windows.Media;
using System.Windows.Shapes;
#endregion
public class WPFWindow : Window
{
	private Canvas canvas = new Canvas();
	public WPFWindow()
	{
		this.AllowsTransparency = true;
		this.WindowStyle = WindowStyle.None;
		this.Background = Brushes.Black;
		this.Topmost = true;
		this.Width = 400;
		this.Height = 300;
		canvas.Width = this.Width;
		canvas.Height = this.Height;
		canvas.Background = Brushes.Black;
		this.Content = canvas;
	}
}
```

Main:

```
WPFWindow w = new WPFWindow();
w.Show();
```

LINKÖPINGS UNIVERSITET

What is the problem with this approach?

How can a designer and a programmer work together?

Do our designers need to learn how to program?

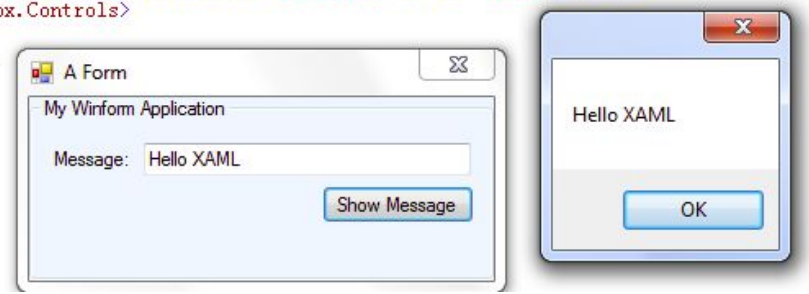
Solution: ***Seperation of Concerns!***

XAML https://msdn.microsoft.com/en-us/library/ms752059(v=vs.110).aspx

XAML is a declarative markup language. As applied to the .NET Framework programming model, XAML simplifies creating a UI for a .NET Framework application. You can create visible UI elements in the declarative XAML markup, and then separate the UI definition from the run-time logic by using code-behind files, joined to the markup through partial class definitions.

Winform.xaml

```
<Form x:Class="XAMLForWinform.Winform"
      xmlns="clr-namespace:System.Windows.Forms;assembly=System.Windows.Forms"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      MaximizeBox="False" MinimizeBox="False" Text="A Form" Width="300" Height="150" BackColor="AliceBlue">
    <Form.Controls>
        <GroupBox x:Name="ctlGroupBox" Text="My Winform Application" Dock="Fill">
            <GroupBox.Controls>
                <Label x:Name="ctlLabel" AutoSize="True" Text="Message:" Location="12, 31" Size="36, 13"/>
                <Button x:Name="ctlButton" Text="Show Message" UseVisualStyleBackColor="True" Location="17
                        Click="Button_Click"/>
                <TextBox x:Name="ctlTextBox" Location="70, 28" Size="200, 20"/>
            </GroupBox.Controls>
        </GroupBox>
    </Form.Controls>
</Form>
```



LINKÖPINGS UNIVERSITET

WPF - sample code (XAML/C#)

WPF https://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx

```
<Window x:Class="WPFWindow" xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Title="MainWindow" Height="350" Width="525">
        <Grid>
                <Button Content="Open Window" Click="ButtonClicked" Height="25" HorizontalAlignment="Left"
        Margin="379,264,0,0" Name="button1" VerticalAlignment="Top" Width="100" />
        </Grid>
</Window>
public class WPFWindow : Window
{
        public MainWindow()

        { InitializeComponent();}

        private void ButtonClicked(object sender,
        RoutedEventArgs e) {

                SubWindow subWindow = new SubWindow();
                subWindow.Show();

        }


}
```

Event handling

```
Main:

WPFWindow w = new WPFWindow();
w.Show();
```

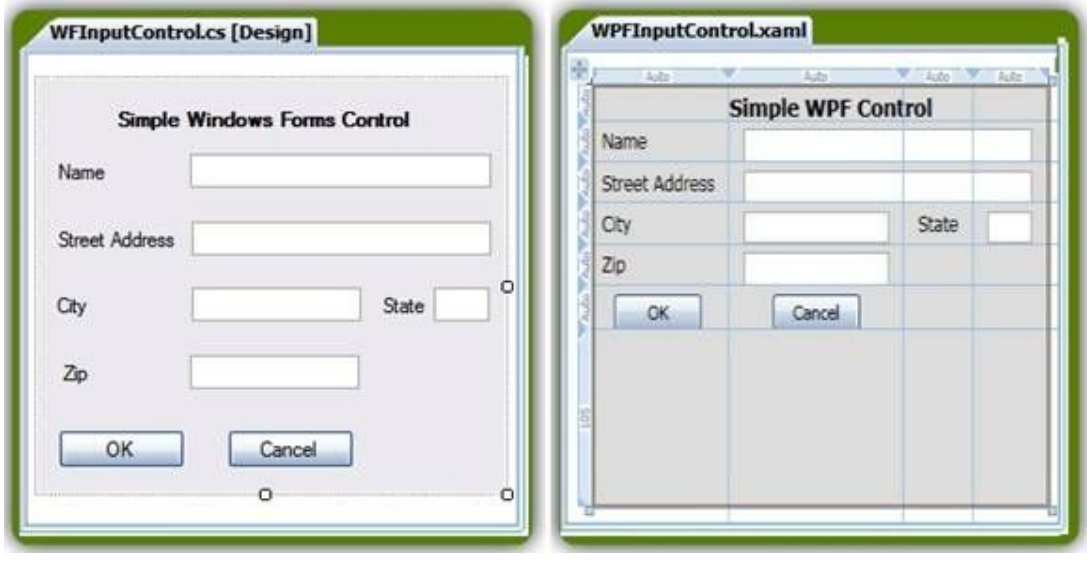
LINKÖPINGS UNIVERSITET

Creating WPF application in visual studio and blend

https://msdn.microsoft.com/en-us/library/ms754130(v=vs.110).aspx

Windows Presentation Foundation (WPF) in Visual Studio 2015 provides developers with a unified programming model for building modern line-of-business desktop applications on Windows.

Blend helps you design top-notch UI look and feel.



LINKÖPINGS UNIVERSITET

Data Binding

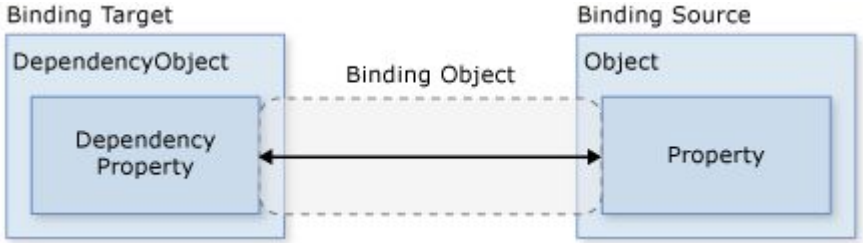
https://msdn.microsoft.com/en-us/library/ms750612(v=vs.110).aspx

Windows Presentation Foundation (WPF) data binding provides a simple and consistent way for applications to present and interact with data. Elements can be bound to data from a variety of data sources in the form of common language runtime (CLR) objects and XML.

Data Binding

https://msdn.microsoft.com/en-us/library/ms750612(v=vs.110).aspx

OneWay binding causes changes to the source property to automatically update the target property, but changes to the target property are not propagated back to the source property. This type of binding is appropriate if the control being bound is implicitly read-only. For instance, you may bind to a source such as a stock ticker or perhaps your target property has no control interface provided for making changes, such as a data-bound background color of a table. If there is no need to monitor the changes of the target property, using the OneWay binding mode avoids the overhead of the TwoWay binding mode.

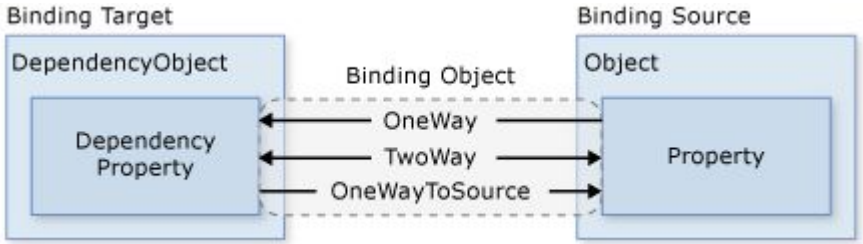


LINKÖPINGS UNIVERSITET

Data Binding

https://msdn.microsoft.com/en-us/library/ms750612(v=vs.110).aspx

TwoWay binding causes changes to either the source property or the target property to automatically update the other. This type of binding is appropriate for editable forms or other fully-interactive UI scenarios. Most properties default toOneWay binding, but some dependency properties (typically properties of user-editable controls such as the Text property of TextBox and the IsChecked property of CheckBox) default to TwoWay binding. A programmatic way to determine whether a dependency property binds one-way or two-way by default is to get the property metadata of the property using GetMetadata and then check the Boolean value of the BindsTwoWayByDefault property.



LINKÖPINGS UNIVERSITET

Explain Binding Mode In WPF

https://www.c-sharpcorner.com/article/explain-binding-mode-in-wpf/

Some demo on binding mode!

LINKÖPINGS UNIVERSITET

Data Binding: How TO

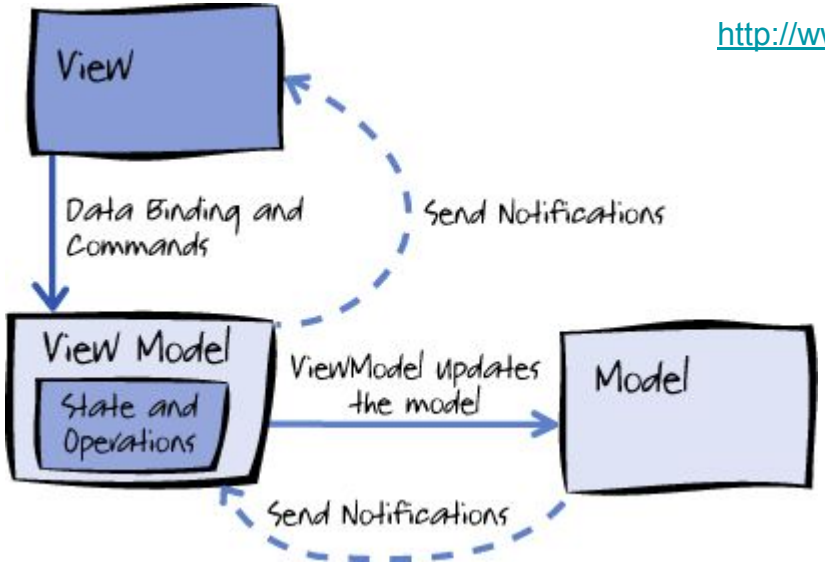
https://blogs.msdn.microsoft.com/jerrynixon/2012/10/12/xaml-binding-basics-101/

LINKÖPINGS UNIVERSITET

Model View ModelView Pattern https://msdn.microsoft.com/en-us/library/hh848246.aspx

The Model-View-ViewModel pattern can be used on all XAML platforms. Its intent is to provide a clean separation of concerns between the user interface controls and their logic.
There are three core components in the MVVM pattern: the model, the view, and the view model. Each serves a distinct and separate role. The following illustration shows the relationships between the three components.

http://www.mindscapehq.com/products/wpfelements/mvvm-pattern-in-wpf



LINKÖPINGS UNIVERSITET

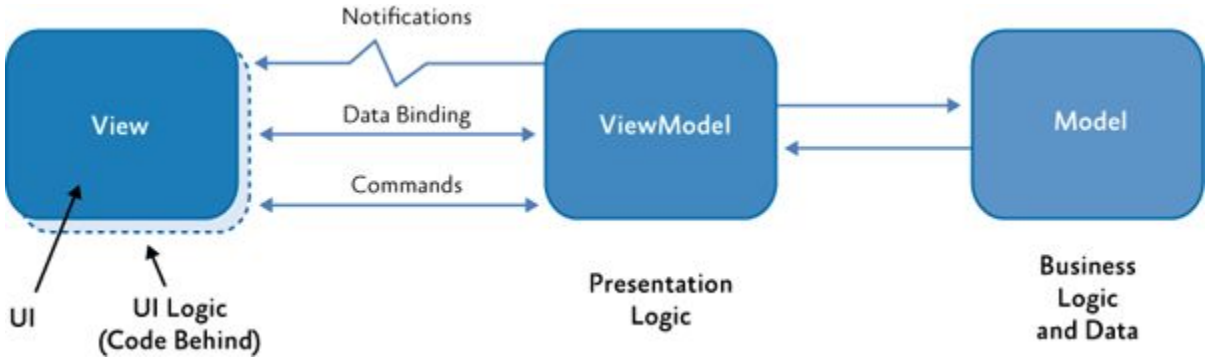
Model View ModelView Pattern https://msdn.microsoft.com/en-us/library/hh848246.aspx

In order for the **view model** to participate in two-way data binding with the view, its properties must raise the **PropertyChanged** event.

LINKÖPINGS UNIVERSITET

MVVM and 3-tier architecture

https://www.codeproject.com/Tips/813345/Basic-MVVM-and-ICommand-Usage-Example



LINKÖPINGS UNIVERSITET

INotifyPropertyChanged Interface

https://docs.microsoft.com/en-us/dotnet/framework/wpf/data/how-to-implement-property-change-notification

- To support OneWay or TwoWay binding to enable your binding target properties to automatically reflect the dynamic changes of the binding source (for example, to have the preview pane updated automatically when the user edits a form), your class needs to provide the proper property changed notifications. This example shows how to create a class that implements INotifyPropertyChanged.

ICommand Interface

https://docs.microsoft.com/en-us/dotnet/api/system.windows.input.icommand?view=netframework-4.7.2

- No need to have handler in the code behind.
- Better decoupling, easier testing.
- ICommand is implemented as part of the presentation logic.
- **Not mandatory in the course.**
- Investigate!

ObservableCollection<T> Class

https://docs.microsoft.com/en-us/dotnet/framework/wpf/data/how-to-create-and-bind-to-an-observablecollection

ObservableCollection is a collection which allows subscribers to be notified when the contents of the collection are altered. This includes replacement of objects, deletion, addition, and movements.

https://www.codeproject.com/Articles/1004644/ObservableCollection-Simply-Explained

LINKÖPINGS UNIVERSITET

2D Graphics

https://msdn.microsoft.com/en-us/library/bb613591(v=vs.110).aspx

- WPF provides both Drawing and Shape objects to represent graphical drawing content.
- Drawing objects are simpler constructs than Shape objects and provide better performance characteristics.
- A Shape allows you to draw a graphical shape to the screen. Because they are derived from the FrameworkElement class, Shape objects can be used inside panels and most controls.

https://www.tutorialspoint.com/wpf/wpf_2d_graphics.htm

LINKÖPINGS UNIVERSITET

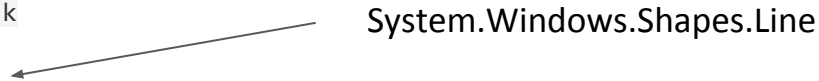
2D Graphics - Sample C# code

https://msdn.microsoft.com/en-us/library/bb613591(v=vs.110).aspx

```
public static void Main()
{
    var app = new Application();
    var window = new Window();
    var canvas = new Canvas();

    window.Content = canvas;
    canvas.Children.Add(new Line
        {
            X1 = 0,
            Y1 = 0,
            X2 = 400,
            Y2 = 400,
            Stroke = Brushes.Black
        });
    canvas.Children.Add(new Line
        {
            X1 = 0,
            Y1 = 400,
            X2 = 400,
            Y2 = 0,
            Stroke = Brushes.Black
        });

    app.Run(window);
}
```



LINKÖPINGS UNIVERSITET

Walkthrough: My First WPF Desktop Application

https://msdn.microsoft.com/en-us/library/ms752299(v=vs.110).aspx

- Defining XAML to design the appearance of the application's user interface (UI).
- Writing code to build the application's behavior.
- Creating an application definition to manage the application.
- Adding controls and creating the layout to compose the application UI.
- Creating styles to create a consistent appearance throughout an application's UI.
- Binding the UI to data to both populate the UI from data and keep the data and UI synchronized.

https://weblogs.asp.net/scottgu/silverlight-tutorial-part-4-using-style-elements-to-better-encapsulate-look-and-feel

LINKÖPINGS UNIVERSITET

Message Dialog Box

https://docs.microsoft.com/en-us/dotnet/framework/wpf/app-development/dialog-boxes-overview

- Display specific information to users.
- Gather information from users.
- Both display and gather information.

Common Dialog Boxes

https://docs.microsoft.com/en-us/dotnet/framework/wpf/app-development/dialog-boxes-overview

Windows implements a variety of reusable dialog boxes that are common to all applications, including dialog boxes for opening files, saving files, and printing.

Font and Color Dialog Boxes

https://www.codeproject.com/Articles/368070/A-WPF-Font-Picker-with-Color

- The Windows Presentation Framework (WPF) comes with no predefined font dialog and color dialog.

WPF Control Library

https://docs.microsoft.com/en-us/dotnet/framework/wpf/controls/control-library

- Windows Presentation Foundation (WPF) ships with many of the common UI components that are used in almost every Windows application.
- You can add a control to an application by using either Extensible Application Markup Language (XAML) or code.
- It is common to change the appearance of a control to fit the look and feel of your application.

https://docs.microsoft.com/en-us/dotnet/framework/wpf/controls/

LINKÖPINGS UNIVERSITET

WPF vs Windows Forms

https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/windows-forms-controls-and-equivalent-wpf-controls

- Many Windows Forms controls have equivalent WPF controls, but some Windows Forms controls have no equivalents in WPF.
- It’s not allowed to use Windows Forms in your solution for the labs.

LINKÖPINGS UNIVERSITET

Styling and Templating

https://docs.microsoft.com/en-us/dotnet/framework/wpf/controls/styling-and-templating

Windows Presentation Foundation (WPF) styling and templating refer to a suite of features (styles, templates, triggers, and storyboards) that allow developers and designers to create visually compelling effects and to create a consistent appearance for their product.

WPF Window

https://docs.microsoft.com/en-us/dotnet/framework/wpf/app-development/wpf-windows-overview

- Users interact with Windows Presentation Foundation (WPF) standalone applications through windows.
- The primary purpose of a window is to host content that visualizes data and enables users to interact with data.
- Standalone WPF applications provide their own windows by using the Window class.

WPF Application

https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/walkthrough-my-first-wpf-desktop-application

- Use XAML to design the appearance of the application's user interface (UI).
- Write code to build the application's behavior.
- Create an application definition to manage the application.
- Add controls and create the layout to compose the application UI.
- Create styles for a consistent appearance throughout an application's UI.
- Bind the UI to data to both populate the UI from data and keep the data and UI synchronized.

Thanks for listening!

LINKÖPINGS UNIVERSITET