WPF Interview question.

<https://www.codeproject.com/Articles/744082/WPF-Interview-questions-with-answers#WhatisWPF>

## What is WPF?

WPF (Windows Presentation foundation) is a graphical subsystem for displaying user interfaces, documents, images, movies etc in windows application.

What is the need of WPF when we had windows forms?

Hide   Copy Code

Remember: - ABCDEFG

A - Anywhere execution ( Windows or Web)

B - Bindings ( less coding)

C - Common look and feel ( resource and styles)

D - Declarative programming (XAML)

E - Expression blend animation ( Animation ease)

F - Fast execution ( Hardware acceleration)

G - Graphic hardware independent ( resolution independent)

## What is XAML in WPF and why do we need it?

XAML is a XML file which represents your WPF UI. The whole point of creating the UI representation in XML was write once and run it anywhere. So the same XAML UI can be rendered as windows application with WPF and the same UI can be displayed on the browser using WPF browser or Silverlight application.

## What is xmlns in XAML file?

“xmlns” stands for XML namespaces. It helps us to avoid name conflicts and confusion in XML documents.

What is the difference between xmlns and xmlns:x in WPF ?

Bothe namespaces helps to define / resolved XAML UI elements.

The first namespace is the default namespace and helps to resolve overall WPF elements.

Hide   Copy Code

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

The second namespace is prefixed by “x:” and helps to resolve XAML language definition.

Hide   Copy Code

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

## Provide some instances where you have “xmlns:x” namespace in XAML ?

There are two common scenarios where we use “xmlns:x” namespace :-

To define behind code for the XAML file using “x:class” attribute.

## When should we use “x:name” vs “name” ?

There is no difference between “x:name” and “name” , “name” is short hand of “x:name”. But in classes where you do not find “name” property ( this is a rare situations) we need to use “x:name” property.

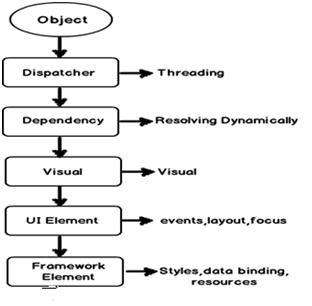
## What are the different kinds of controls in WPF?

WPF controls can be categorized in to four categories:-

* **Control: -** This is the basic control with which you will work most of time. For example textbox, buttons etc. Now controls which are standalone control like button , text box , labels etc are termed as content control. Now there are other controls which can hold other controls, for instance itemscontrols. Itemscontrol can have multiple textbox controls, label controls etc.
* **Shape: -** These controls help us to create simple graphic controls like Ellipse, line, rectangle etc.
* **Panel: -** These controls help to align and position the controls. For instance grid helps us to align in a table manner, stack panel helps for horizontal and vertical alignment.
* **Content presenter: -** This control helps to place any XAML content inside it. Used when we want to add dynamic controls on a WPF screen.

All the above four types of WPF controls finally inherit from the frameworkelement class of WPF.

## Can you explain the complete WPF object hierarchy?



## So is XAML meant only for WPF ?

No,XAML is not meant only for WPF.XAML is a XML-based language and it had various variants.

**WPF XAML** is used to describe WPF content, such as WPF objects, controls and documents. In WPF XAML we also have XPS XAML which defines an XML representation of electronic documents.

**Silverlight XAML** is a subset of WPF XAML meant for Silverlight applications. Silverlight is a cross-platform browser plug-in which helps us to create rich web content with 2-dimensional graphics, animation, and audio and video.

**WWF XAML** helps us to describe Windows Workflow Foundation content. WWF engine then uses this XAML and invokes workflow accordingly.

## Can you explain the overall architecture of WPF?

**User32:-** It decides which goes where on the screen.

**DirectX: -** As said previously WPF uses directX internally. DirectX talks with drivers and renders the content.

**Milcore: -** Mil stands for media integration library. This section is a unmanaged code because it acts like a bridge between WPF managed and DirectX / User32 unmanaged API.

**Presentation core :-** This is a low level API exposed by WPF providing features for 2D , 3D , geometry etc.

**Presentation framework:-** This section has high level features like application controls , layouts . Content etc which helps you to build up your application.

## What is App.xaml in WPF project?

App.xaml is the start up file or a boot strapper file which triggers your first XAML page from your WPF project.

## What are various ways of doing alignment in WPF?

There are five ways of doing alignment in WPF:- Grid, Stack Panel, WrapPanel, Dock Panel, Canvas/

## What are resources in WPF?

Resources are objects referred in WPF XAML. It could be shared by items.

For example, define a Windows.Resources as SloldColorBrush Color and is referred by Texbox Background or button background color.

Explain the difference between static and dynamicresource?

Resources can be referred statically or dynamically. Static referred resources evaluate the resource only once and after that if the resources change those changes are not reflected in the binding. While dynamic referred resources are evaluated every time the resource is needed.

Consider the below “SolidColorBrush” resource which is set to “LightBlue” color.

Hide   Copy Code

<Window.Resources>

<SolidColorBrush Color="LightBlue" x:Key="buttonBackground" />

</Window.Resources>

The above resource is binded with the below two textboxes statically and dynamically respectively.

Hide   Copy Code

<TextBox Background="{DynamicResource buttonBackground}" />

<textbox background="{StaticResource buttonBackground}">

Now if we modify the resource , you see the first text box changes the background and the other textbox color stays as it is.

Hide   Copy Code

private void Button\_Click(object sender, RoutedEventArgs e)

{

Resources["buttonBackground"] = Brushes.Black;

}

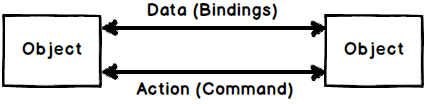
Only DynamicResource item color is changed.

## When should we use static resource over dynamic resource ?

Dynamic resources reduce application performance because they are evaluated every time the resource is needed. So the best practice is use Static resource until there is a specific reason to use dynamic resource. If you want resource to be evaluated again and again then only use dynamic resource.

## Explain the need of binding and commands?

WPF Binding’s helps to send / receive data between WPF objects while command helps to send and receive actions.The object that emits data or action is termed as **source** and the object who wants to receive data or action is termed as **target**.



## Explain one way, two way, one time and one way to source?

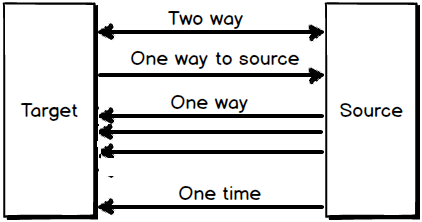
All the above 4 things define how data will flow between target and source objects when WPF binding is applied.

**Two way: -** Data can flow from both source to target and from target to source.

**One way: -** Data flows only from source to target.

**One way to source: -** Data flows only from target to source.

**One time: -** Data flows only for the first time from source to target and after that no communication happens.



Can you explain WPF command with an example?

When end users interact with application they send actions like button click, right click , control + c, control + v etc. A command class in WPF wraps these end user actions in to a class so that they can be reused again and again.

WPF Command class idea is an implementation of command pattern from gang of four design pattern.

To create a command class we need to implement the “ICommand” interface. For example below is a simple command class which increments a counter class by calling “Increment” method.

Hide   Copy Code

public class IncrementCounter : System.Windows.Input.ICommand

{

private clsCounter obj;

public IncrementCounter(clsCounter o)

{

obj = o;

}

public bool CanExecute(object parameter)

{

return true;

}

public event EventHandler CanExecuteChanged;

public void Execute(object parameter)

{

obj.Increment();

}

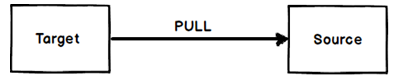
}

## How does “UpdateSourceTrigger” affect bindings?

“UpdateSourceTrigger” decides when the data should get updated between WPF objects that are binded. In other word should data get updated in lost focus event, in data change event etc.

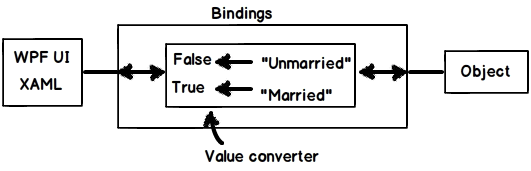
## Explain the need of “INotifyPropertyChanged” interface?

When we bind two WPF objects the target data is updated depending on the “UpdateSourceTrigger” events. Please refer the previous question for “UpdateSourceTrigger” basics.  
The “UpdateSourceTrigger” has events like lostfocus , property change etc. In other words when lostfocus or property change event happen on the target it makes a PULL to the source to get the latest data.



## What are value converters in WPF?

Binding is one of the big features in WPF which helps us to facilitate data flow between WPF UI and Object. But when data flows from source to UI or vice-versa using these bindings we need to convert data from one format to other format. For instance let’s say we have a “Person” object with a married string property.



## Explain multi binding and multivalue converters in WPF?

## “MultiBinding” helps you bind multiple sources to a target while multi-converters acts like bridge if the source and targets have different data formats or need some conversion.

Explain WPF relative binding / relative resource?

When we define bindings we need at least two elements target and source. But many times rather than defining binding between two elements we would like to define binding with reference to the current element i.e. RELATIVELY.

For instance let’s say we have a WPF border and we would like height and width of the broder to be same. So for this scenario the target and source are the same, the WPF border itself. So we can define the binding using “RelativeSource” as shown in the below code. You can see it uses “Self” binding mode to bind the element to itself.

Hide   Copy Code

<Border BorderBrush="Black" BorderThickness="1" Height="139"

Width="{Binding Height, RelativeSource={RelativeSource Self}}"/>

## What are the different ways of binding using relative source?

There are four ways of binding relatively in WPF :-

Self

Ancestor

Previousdata

Templated parent

## Can you explain self relative source binding in WPF?

This relative binding helps to bind to one property of an element to the other property of the same element. For example in the below XAML the border width is binded to height of the same border element.

Hide   Copy Code

<Border BorderBrush="Black" BorderThickness="1" Height="139"

Width="{Binding Height, RelativeSource={RelativeSource Self}}"/>

Explain Ancestor relative source binding in WPF?

This relative binding helps to bind properties to the parent element properties. For example in the below XAML code we have a textbox which has two border’s as a parent. One border is having dark green and the other border is having dark red color as the border color.

The dark green color border is the parent element followed by dark red and the text box the child element at the end of the hierarchy.

Hide   Copy Code

<Border BorderBrush="DarkGreen"> <!-- *Level 2 ->*

<Border BorderBrush="DarkRed"> *<!-- Level 1 ->*

<TextBox *removed="{Binding BorderBrush, RelativeSource={RelativeSource FindAncestor, AncestorLevel=1, AncestorType= {x:Type Border}}}"*/>

</Border>

</Border>

Explain the difference between visual and logical tree in WPF ?

WPF UI is represented in XAML which is a XML format. In XML elements are arranged in a hierarchal fashion. For example if you look at the below XAML (it has been downsized for simplicity) we have a Window element, the window element has a Grid control and the grid control has a button inside it.

Hide   Copy Code

<Window>

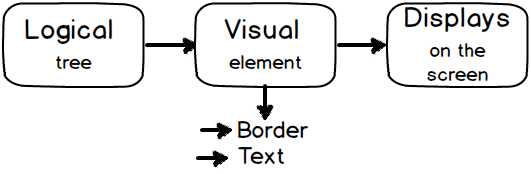
<Grid>

<Button..>

</Grid>

</Window>

So if you visualize the above XAML **logically** you can think that the button control is a child element of the Grid and the grid is the child element of the Window. This relationship between the elements which looks logical looking at the XAML is termed as “**Logical tree**”.



But now to display this Logical tree on to your screen you need lot of visual elements. like border, text etc. So when you add these visual elements to the logical tree that complete structure is termed as “**Visual Tree**”.

## Why do we need to have these perspective of visual and logical tree in WPF ?

Visual tree and Logical tree are important when you work with WPF routed events.

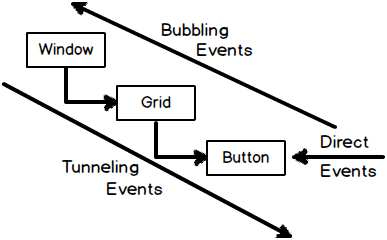
## Explain routed events in WPF?

Routed events are those events which travel up or down the visual tree hierarchy. WPF events can be classified in to 3 types:-

Direct events: - In this case event is raised at the source and handled at the source itself like “MouseEnter” events.

Bubbling events: - They travel up the visual tree hierarchy. For example “MouseDown” is a bubbling event.

Tunneling events: - These events travel down the visual tree hierarchy. “PreviewKeyDown” is a tunneling event.



What are styles in WPF?

WPF style helps to define look and feel ( color , fonts, alignments etc) from a central location

## What are style triggers?

Many times we would want a style property to execute under certain conditions.

## <Style x:Key="myStyle" TargetType="Button">

## <Style.Triggers>

## <Trigger Property="IsFocused" Value="True">

## <Setter Property="Background" Value="Red"/>

## </Trigger>

## </Style.Triggers>

## <Setter Property="Background" Value="Aqua" />

## </Style>

## What is MVVM?

MVVM is an architecture pattern where we divide the project in to three logical layers and every layer has its own responsibility.

What are the benefits of MVVM?

Separation of concern, Increased UI Reusability, Automated UI Unit Testing.

What is the importance of command and bindings in MVVM pattern?

MVVM in the most used architecture because of command and bindings facility provided by WPF.

What is the difference between MVVM and 3 layer architecture?

*MVVM has extra layer view model of View and Model*

Explain delegate command?

*Delegate command makes a MVVM command class independent of the view mode*

<Button Name="colbutton" Command="{Binding UpdataCommand}">Color</Button>

<Button Name="updatebutton" Command="{Binding ChangecolorCommand}" >Update</Button>

public ICommand UpdataCommand

{

get

{

return new DelegatingCommand(Clear); //abstract pointer to function/methods to DelegatingCommand

}

}

public ICommand ChangecolorCommand

{

get

{

return new DelegatingCommand(AddName); //abstract pointer to function/methods to DelegatingCommand

}

}

class DelegatingCommand : ICommand

{

//backing fields hold the delegates! To decouple the view model class from command we can use delegates

// i.e. "Action" and "Func". The command class only need two things "WhattoExecute" and "WhentoExute".

// just passed these methods as generic delegates (abstract pointer to function/methods).

private readonly Action<object> \_action;

private readonly Func<object, bool> \_canExecute;

//private CustomerViewModel \_viewModel;

public DelegatingCommand(Action action)

: this((o) => action())

{ }

public DelegatingCommand(Action<object> action)

: this(action, (o) => true)

{ }

public DelegatingCommand(Action<object> action, Func<object, bool> canExecute)

{

\_action = action;

this.\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return \_canExecute(parameter);

}

public event System.EventHandler CanExecuteChanged{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public void Execute(object parameter)

{

this.\_action(parameter);

}

}

What is PRISM?

PRISM is a framework to develop composite application in WPF and Silverlight.

Prebuilt components, assemble them together and create the application.

What are benefits of PRISM?

Developing components as independent units then running through same interface. s the components are developed in individual units we can plug them using PRISM and create composed UI in an easy way.

How are individual units combined in to a single unit?

PRISM uses dependency injection for the same.

Does PRISM do MVVM?

The prime focus of PRISM is modular development and not MVVM. But it has delegate command which can reduce MVVM codes.

Is PRISM a part of WPF?

Prism is the [Microsoft Patterns and Practices Team](http://msdn.microsoft.com/en-us/practices/bb190332) official guidance for building "composite applications" in WPF and Silverlight.

Its intended to provide guidance on the best practices for building large scale applications which are flexible in terms of development and maintainability.

This includes guidance on dependency injection (via Unity or MEF), layout (including using MVVM), composite event handling, etc.

No, PRISM is a separate installation

What is expression blend?

**Expression Blend**. **Expression Blend** is part of **Microsoft's Expression** Studio suite of **design tools**. It is the production **tool** for **designing** and building user interfaces for **Silverlight** on the **Web** and Windows Phone, as well as for **WPF** (Windows Presentation Foundation) on Windows and **Microsoft**