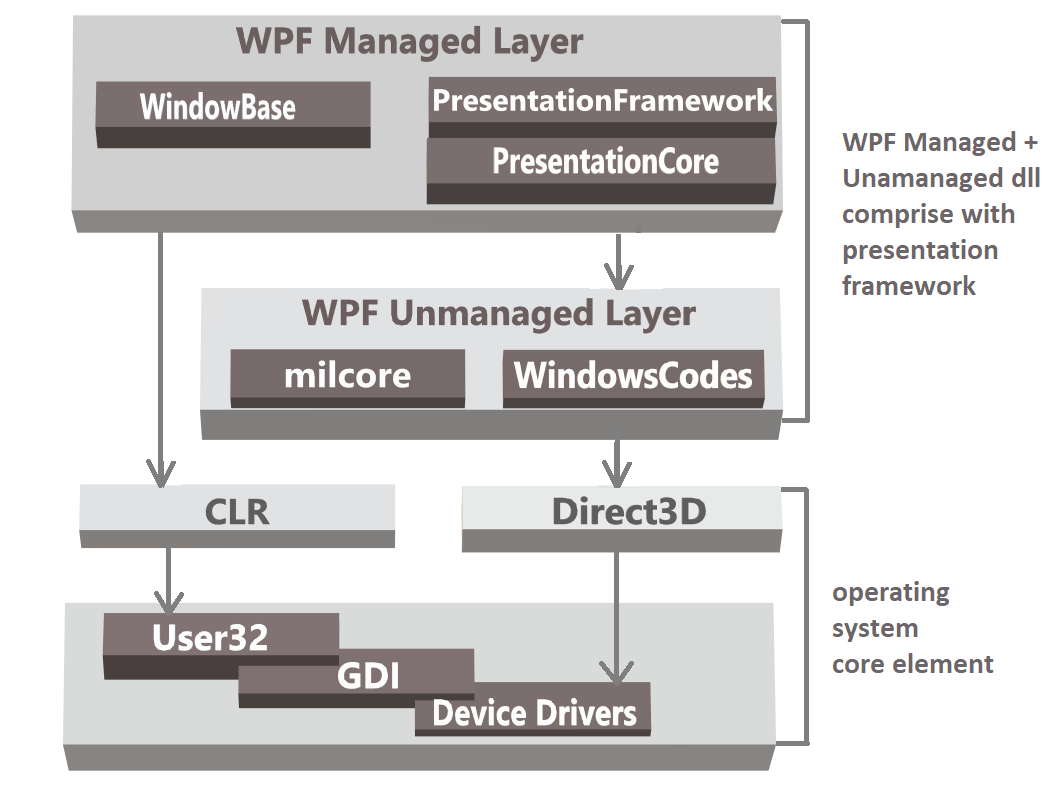
**WPF in .NET**

The Windows Presentation Foundation (WPF) is Microsoft’s “next generation UI framework” to create applications with a rich user experience. It is part of the .NET framework 3.0 and higher.

**Why It Is Used?**

With Windows forms application it is not possible to use Advance graphics and window form will always be inefficient in comparison to WPF.

**Architecture of WPF**

****

**Feature of WPF**

* XAML as the Markup: XAML goes in some ways beyond XML because it is simple to understand.
* It provide dependency properties such as data binding, styling, triggers, and animation.
* Styles & Triggers are very useful in CSS
* Control templates: Separation of appearance and behavior
* Integration of 2D and 3D

**Advantages**

The advantages of WPF applications include:

**Control rotation**

* Windows Forms applications support only text rotation whereas WPF support control rotation.

**Culturally aware controls**

* Static text in controls and the return data from String function are modified according to the culture and language specified by user's operating system.

**Development Tools**

* Microsoft provides two development tools for WPF applications.
* One is Visual Studio which is made for developers ,
* The other is Expression Blend, which is made for designers. While Visual Studio is good in code and XAML editing, it has a rare support for all the graphical stuff like gradients, template editing, animation, etc. This is the point where Expression Blend comes in. Blend covers the graphical part very well but it has (still) rare support for code and XAML editing.

**Hardware acceleration**

* WPF is built on top of Direct3D, which offloads work to graphics processing units (GPUs) instead of central processor units (CPUs).
* This provides hardware acceleration that permit smoother graphics and enhanced performance.

**Programming through XAML**

Benefits of XAML are as below:

* XAML code is short and clear to read
* Concise implementation (Similar to HTML)
* Human Readable (except Vector & Meshes)
* Can be rendered in the browser / standalone application
* Interoperable between multiple tools such as Blend, Orcas, XAMLPAD
* This allows graphic designers to directly contribute to the look and feel of WPF applications.

**Resolution Independent**

* WPF lets you shrink or enlarge elements on the screen, independent of the screen’s resolution. All measures in WPF are logical units - not pixels.
* If you increase the resolution of your screen, the user interface stays in the same size - it just gets crispier.
* Since WPF builds on a vector based rendering engine it is incredibly easy to build saleable user interfaces.

**Rich composition and customization**

* WPF controls are easily customizable.
* WPF allow you to create skins for applications that have radically different looks.

**Separations of behavior and appearance**

* WPF separates the appearance of a user interface from its behavior. The appearance is generally specified in the Extensible Application Markup Language (XAML), the behavior is implemented in a managed programming language like C# or Visual Basic.
* Data binding, events and commands tie the two parts together. The separation of appearance and behavior brings the following benefits:
  + Designers and developers can work on separate models.
  + Graphical design tools can work on simple XML documents instead of parsing code.
  + Appearance and behavior are loosely coupled

**Tight multimedia integration**

* To use 3-D graphics, video, speech, and rich document viewing in Windows Forms applications, you would need to learn several independent technologies and blend them together without much built-in support.
* WPF applications allow you to use all these features with a consistent programming model.

**Disadvantage**

* It has less controls as compare to web forms
* Not Compatible with older version

**Types of WPF Application**

* + Windows Application
  + Navigation Application
  + XAML Browser Application(XBAPs)

**Windows Application**

* Windows Application is similar to Windows Form Application
* We can open multiple windows at any given time, and there is no built-in sense of navigation or history
* When We Use windows application?
  + We use windows application for a user experience that closely resembles a traditional Windows Form application.
  + In menu driven, multiwindow application that combines the rich functionality of a desktop with the rich UI.

**XBAPs Application**

* XAML Browser Application
* XBAPs application are similar to Navigation Applications, but they are designed to run in Windows Internet Explorer.
* XBAPs run under a partial-trust environment.

When we Use XBAPs application?

* + When you want to deploy the application to a web server, which will start from the hyperlink. It will make it easily accessible to a large-scale audience.
  + If your application does not require access to system resources, XBAP might be good choice.

**WPF Assemblies**

Windowbase.dll

* + It defines the core types, which constructs the infrastructure of WPF API.

Presentationcore.dll

* + This assembly defines numerous types that construct the foundation of the WPF GUI layer.

PresentationFoundation.dll

* + It defines WPF control types, animation & multimedia support, data binding support and other WPF services.

**XAML vs. Code**

In this example we will create simple stackpanel with textbox and button in XAML and C#

**In XMAL**

 <StackPanel>

<TextBlock Margin="20">Welcome to the World of XAML</TextBlock>

<Button Margin="10"HorizontalAlignment="Right">OK</Button>

**In C#**

StackPanel stackPanel = [**new**](http://www.google.com/search?q=new+msdn.microsoft.com)StackPanel ();

**this**.Content = stackPanel;

// Create the TextBlock

TextBlock textBlock = [**new**](http://www.google.com/search?q=new+msdn.microsoft.com) TextBlock ();

textBlock.Margin = [**new**](http://www.google.com/search?q=new+msdn.microsoft.com) Thickness (10);

textBlock.Text = "Welcome to the World of XAML";

stackPanel.Children.Add(textBlock);

// Create the Button

Button button = [**new**](http://www.google.com/search?q=new+msdn.microsoft.com) Button();

button.Margin= [**new**](http://www.google.com/search?q=new+msdn.microsoft.com) Thickness(20);

button.Content = "OK";

stackPanel.Children.Add(button);

 As you can see, XAML version is shorter and clearer to read which is power of XAMLs expressiveness.