One of the first things you will encounter while working with WPF is XAML. XAML stands for Extensible Application Markup Language. It’s a simple and declarative language based on XML.

* In XAML, it very easy to create, initialize, and set properties of objects with hierarchical relations.
* It is mainly used for designing GUIs, however it can be used for other purposes as well, e.g., to declare workflow in Workflow Foundation.

Basic Syntax

When you create your new WPF project, you will encounter some of the XAML code by default in MainWindow.xaml as shown below.

<Window x:Class = "Resources.MainWindow"

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>

</Grid>

</Window>

The above XAML file contains different kinds of information. The following table briefly explains the role of each information.

|  |  |
| --- | --- |
| **Information** | **Description** |
| <Window | It is the opening object element or container of the root. |
| x:Class = "Resources.MainWindow" | It is a partial class declaration which connects the markup to the partial class code defined behind. |
| xmlns = ["http://schemas.microsoft.com/win fx/2006/xaml/presentation"](http://schemas.microsoft.com/win%20fx/2006/xaml/presentation) | Maps the default XAML namespace for WPF client/framework |
| xmlns:x = ["http://schemas.microsoft.com/w infx/2006/xaml"](http://schemas.microsoft.com/w%20infx/2006/xaml) | XAML namespace for XAML language which maps it to x: prefix |
| > | End of object element of the root |
| <Grid>  </Grid> | It is starting and closing tags of an empty grid object. |
| </Window> | Closing the object element |

The syntax rules for XAML is almost similar to XML. If you look at an XAML document, then you will notice that it is actually a valid XML file, but an XML file is not necessarily an XAML file. It is because in XML, the value of the attributes must be a string while in XAML, it can be a different object which is known as Property element syntax.

* The syntax of an Object element starts with a left angle bracket (<) followed by the name of an object, e.g. Button.
* Define some Properties and attributes of that object element.
* The Object element must be closed by a forward slash (/) followed immediately by a right angle bracket (>).

Why XAML in WPF

XAML is not only the most widely known feature of WPF, but it's also one of the most misunderstood features. If you have exposure to WPF, then you must have heard of XAML; but take a note of the following two less known facts about XAML −

* WPF doesn't need XAML
* XAML doesn't need WPF

They are in fact separable pieces of technology. To understand how that can be, let's look at a simple example in which a button is created with some properties in XAML.

<Window x:Class = "WPFXAMLOverview.MainWindow"

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

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

Title = "MainWindow" Height = "350" Width = "604">

<StackPanel>

<Button x:Name = "button" Content = "Click Me" HorizontalAlignment = "Left"

Margin = "150" VerticalAlignment = "Top" Width = "75" />

</StackPanel>

</Window>

In case you choose not to use XAML in WPF, then you can achieve the same GUI result with procedural language as well. Let’s have a look at the same example, but this time, we will create a button in C#.

using System.Windows;

using System.Windows.Controls;

namespace WPFXAMLOverview {

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window {

public MainWindow() {

InitializeComponent();

// Create the StackPanel

StackPanel stackPanel = new StackPanel();

this.Content = stackPanel;

// Create the Button

Button button = new Button();

button.Content = "Click Me";

button.HorizontalAlignment = HorizontalAlignment.Left;

button.Margin = new Thickness(150);

button.VerticalAlignment = VerticalAlignment.Top;

button.Width = 75;

stackPanel.Children.Add(button);

}

}

}

<Window x:Class="WpfApplicationDemo.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:WpfApplicationDemo"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid ShowGridLines="True">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200"></ColumnDefinition>

<ColumnDefinition Width="325"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="100"></RowDefinition>

<RowDefinition Height="100"></RowDefinition>

<RowDefinition Height="150"></RowDefinition>

</Grid.RowDefinitions>

<TextBlock Text="Row0 Column0" Grid.Row="0" Grid.Column="0" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>

<TextBlock Text="Row0 Column1" Grid.Row="0" Grid.Column="1" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>

<TextBlock Text="Row1 Column0" Grid.Row="1" Grid.Column="0" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>

<TextBlock Text="Row1 Column1" Grid.Row="1" Grid.Column="1" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>

<TextBlock Text="Row2 Column0" Grid.Row="2" Grid.Column="0" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>

<TextBlock Text="Row2 Column1" Grid.Row="2" Grid.Column="1" FontSize="16" VerticalAlignment="Center" HorizontalAlignment="Center"></TextBlock>

</Grid>

</Window>

<Grid ShowGridLines="True">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

<ColumnDefinition Width="\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="\*"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

<RowDefinition Height="2\*"></RowDefinition>

</Grid.RowDefinitions>

<TextBlock Text="Merging top 3 Cells" Grid.ColumnSpan="3" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="18"></TextBlock>

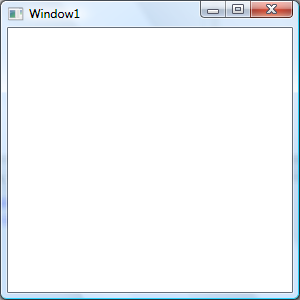
<TextBlock Text="Rowspan=2" Grid.Row="1" Grid.Column="0" Grid.RowSpan="2" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="18"></TextBlock>

<TextBlock Text="Rowspan=2 Colspan=2" Grid.Row="1" Grid.Column="1" Grid.RowSpan="2" Grid.ColumnSpan="2" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="18"></TextBlock>

</Grid>

Proper layout and positioning are vital parts of interactive, high-performance and user-friendly Windows applications.

**Panels**  
  
All Panel controls are defined in the System.Windows.Controls namespace that resides in presentationframework.dll assembly. Besides the root Window, a Panel is the base control that works as a parent control for other child controls. If you create a WPF application the default code for the Window looks as in figure below, where you can see a Grid panel is the default parent control for child controls that may be placed on a Window.



WPF comes with the following five built-in panels:

* Canvas
* DockPanel
* Grid
* StackPanel
* WrapPanel

The purpose and use of these panels is different. Each panel has a different way to position and reposition child controls placed within that panel. The following articles in this series will summarise these panels and their usages.  
  
Similar to any other WPF control, a Panel control may be represented in two ways. First, at design-time using XAML elements and attributes, and second, at run-time, using a WPF class and its properties.

<Grid Name="GridPanel" Background="Blue"

Width="250" Height="200"

VerticalAlignment="Top"

HorizontalAlignment="Left"

FlowDirection="LeftToRight"

/>

Through coding

private void CreateDynamicPanel()

{

// Create a Grid Panel control

Grid gridPanel = new Grid();

// Set Grid Panel properties

gridPanel.Background = new SolidColorBrush(Colors.Blue);

gridPanel.Width = 250;

gridPanel.Height = 200;

gridPanel.HorizontalAlignment = HorizontalAlignment.Left;

gridPanel.VerticalAlignment = VerticalAlignment.Top;

gridPanel.FlowDirection = FlowDirection.LeftToRight;

// Set Grid Panel as content of the Window

RootWindow.Content = gridPanel;

}

STackPanel

<StackPanel>

<TextBlock Margin="10" FontSize="20">How do you like your coffee?</TextBlock>

<Button Margin="10">Black</Button>

<Button Margin="10">With milk</Button>

<Button Margin="10">Latte machiato</Button>

<Button Margin="10">Chappuchino</Button>

</StackPanel>

DockPanel

<DockPanel LastChildFill="True">

<Button Content="Dock=Top" DockPanel.Dock="Top"/>

<Button Content="Dock=Bottom" DockPanel.Dock="Bottom"/>

<Button Content="Dock=Left"/>

<Button Content="Dock=Right" DockPanel.Dock="Right"/>

<Button Content="LastChildFill=True"/>

</DockPanel>

<DockPanel LastChildFill="True">

<Button Content="Dock=Left"/>

<Button Content="Dock=Left"/>

<Button Content="Dock=Top" DockPanel.Dock="Top"/>

<Button Content="Dock=Bottom" DockPanel.Dock="Bottom"/>

<Button Content="Dock=Right" DockPanel.Dock="Right"/>

<Button Content="LastChildFill=True"/>

</DockPanel>

Data Binding

using System;

using System.Collections.Generic;

using System.Text;

namespace WpfApplicationDemo

{

public class Person

{

private string nameValue;

public string Name

{

get { return nameValue; }

set { nameValue = value; }

}

private double ageValue;

public double Age

{

get { return ageValue; }

set

{

if (value != ageValue)

{

ageValue = value;

}

}

}

}

}

Step 2:

public partial class MainWindow : Window

{

**Person person = new Person { Name = "Salman", Age = 26 };**

public MainWindow()

{

InitializeComponent();

**this.DataContext = person;**

}

Step 3:

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

{

string message = person.Name + " is " + person.Age;

MessageBox.Show(message);

}

Xaml file

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height = "Auto" />

<RowDefinition Height = "Auto" />

<RowDefinition Height = "\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width = "Auto" />

<ColumnDefinition Width = "200" />

</Grid.ColumnDefinitions>

<Label Name = "nameLabel" Margin = "2">\_Name:</Label>

<TextBox Name = "nameText" Grid.Column = "1" Margin = "2"

Text = "{Binding Name, Mode = OneWay}"/>

<Label Name = "ageLabel" Margin = "2" Grid.Row = "1">\_Age:</Label>

<TextBox Name = "ageText" Grid.Column = "1" Grid.Row = "1" Margin = "2"

Text = "{Binding Age, Mode = OneWay}"/>

<StackPanel Grid.Row = "2" Grid.ColumnSpan = "2">

<Button Content = "\_Show..." Click="Button\_Click" />

</StackPanel>

</Grid>