**Control Elements**

**Control Element 에는**

**버튼(buttons),체크박스 (check boxes), 라디오버튼(radio buttons),리스트박스 (list boxes), 콤보박스(combo boxes), 메뉴(menus),스크롤바 (scroll bars), 슬라이더(sliders), 기타등등 이 있다.**

**item control 들은 자식컨드롤을 가지고있는 집합체이다.**

**헤더 아이템 컨트롤은 Content 속성을 가지지않지만 헤더 와 아이템 속성(item attributes)을 지니고있다.**

**[Button]**

**<Button Click="ButtonClicked">Button</Button>**

**[radio buttons]**

**<StackPanel>**

**<RadioButton>To be</RadioButton>**

**<RadioButton>Not to be</RadioButton>**

**</StackPanel>**

**[Slider and Scroll Controls]**

**■ ScrollViewer**

   내부 엘리먼트의 크기가 오버플로되면 스크롤바를 생성해준다

|  |  |
| --- | --- |
| **[XAML]** | <!-- 크기가 초과되면 스크롤바 생성, 너비/높이 300 -->  <ScrollViewer Width="300" Height="300"  VerticalScrollBarVisibility="Auto">  <StackPanel>  <StackPanel.Children>  <Button>버튼</Button>  <Button>버튼</Button>  <Button>버튼</Button>  <Button>버튼</Button>  <Button>버튼</Button>  <Button>버튼</Button>  <Button>버튼</Button>  <Button>버튼</Button>  <Button>버튼</Button>  </StackPanel.Children>  </StackPanel>  </ScrollViewer> |
| **[실행]** | http://postfiles8.naver.net/20130227_263/curlicu_1361891423740LOpJk_PNG/022613_1510_WPF1.png?type=w2 |

**[ProgressBar]**

**[Text Controls]**

**<StackPanel Orientation="Horizontal">**

**<TextBox Margin="5" VerticalAlignment="Center" Text="Single line textbox" />**

**<TextBox AcceptsReturn="True" Margin="5" Height="50"**

**VerticalScrollBarVisibility="Visible"**

**VerticalAlignment="Center" Text="Multiline textbox" />**

**<PasswordBox Margin="5" VerticalAlignment="Center" Password="Un5ecure" />**

**</StackPanel>**

**[Label]**

**<Label>label</Label>**

**<TextBox Width="100">**

**<TextBox.ToolTip>**

**<ToolTip Content="Type something here" />**

**</TextBox.ToolTip>**

**</TextBox>**

**[List Controls]**

**<ComboBox>**

**<Button>Click!</Button>**

**<TextBlock>Hello, world</TextBlock>**

**<StackPanel Orientation="Horizontal">**

**<TextBlock>Ellipse:</TextBlock>**

**<Ellipse Fill="Blue" Width="100" />**

**</StackPanel>**

**</ComboBox>**

**TextBlock**은 Text를 다루는 컨트롤 중 가장 빠르고 간단한 형태를 지니고 있다. (그렇다고 지원 요소가 적은 것은 아니다.) TextBlock은 Text를 다루는 강력하고 빠른 컨트롤이라고 이해하는 것이 좋을 것이다. 아래 예제 소스를 살펴보자

**예제1) Textblock**

<Window x:Class="\_1110.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">

<StackPanel>

<TextBlock Name="textBlock1" TextWrapping="Wrap">

<Bold>TextBlock</Bold> 은 <Italic>lightweight</Italic> 에의해 디자인되고,

통합되어 설계된다.

</TextBlock>

<Button Width="100" Margin="10">Click Me</Button>

<TextBlock x:Name="textBlock2"

TextWrapping="Wrap" Background="AntiqueWhite" TextAlignment="Center"

><Run Language="ko-kr" Text="TextBlock"/><LineBreak/><Run Language="ko-kr"/></TextBlock>

<Button Width="100" Margin="10">Click Me</Button>

</StackPanel>

</Window>

**예제2) ContentControls**

<Window x:Class="asdfawefwqer.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">

<StackPanel

Name="HelloWorldGrid"

Background="AntiqueWhite"

>

<Button Height="30" Width="150" Background="Transparent">

<TextBox>Click to type in here</TextBox>

</Button>

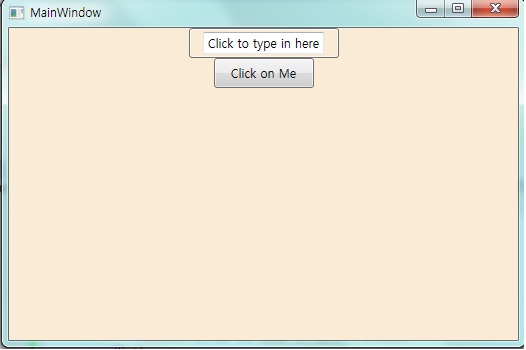
<Button Height="30" Width="100">

Click on Me

</Button>

</StackPanel>

</Window>



**예제2)결과창**

**예제3) Dock\_Panel**

각 지정 방향에 자식 요소를 배치한다. 자식 요소는 겹칠 수 없다.

<Window x:Class="asdfawefwqer.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 Name="StuffToBuy">

<DockPanel>

<Image Source="http://www.oreilly.com/images/oreilly/add\_to\_cart.gif" Height="33" Width="142" />

<Button Height="30" Width="150" Background="Transparent">

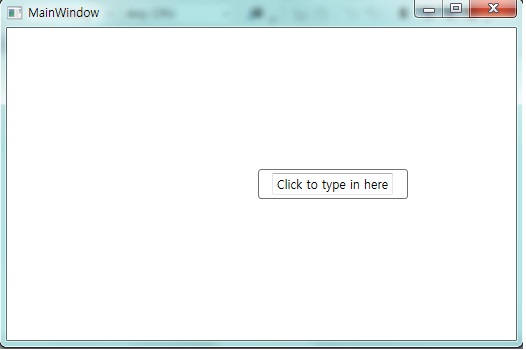
<TextBox>Click to type in here</TextBox>

</Button>

</DockPanel>

</Grid>

</Window>

****

**예제3)결과창**

**예제4) Stack\_Panel**

가로나 세로 방향으로 자식 요소를 일렬로 정렬한다. 자식요소는 겹칠 수 없다.

**StackPanel 의 장점과 단점**

- Grid만큼 유연하지 않지만(원래 StackPanel은 작게 구분된 영역을 정렬하는데 유용하다.)  Style을 설정함으로써 Grid와

  비슷한 유연함을 가질 수 있다.

- 기존 UI 엘리먼트의 변경, 새로운 엘리먼트 추가 시 번호를 매기는 번거로움이 없다.

<Window x:Class="asdfawefwqer.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 Name="StuffToBuy">

<StackPanel>

<Image Source="http://www.oreilly.com/images/oreilly/add\_to\_cart.gif" Height="33" Width="142" />

<Button Height="30" Width="150" Background="Transparent">

<TextBox>Click to type in here</TextBox>

</Button>

</StackPanel>

</Grid>

</Window>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

ex\_1();

}

public void ex\_1()

{

StackPanel Stack = new StackPanel();

this.AddChild(Stack);

//------------

Image img = new Image();

BitmapImage myImg = new BitmapImage();

myImg.BeginInit();

myImg.UriSource = new Uri("http://www.oreilly.com/images/oreilly/add\_to\_cart.gif", UriKind.Absolute);

myImg.DecodePixelWidth = 200;

myImg.EndInit();

img.Source = myImg;

Stack.Children.Add(img);

//----------------

Button btn = new Button();

btn.Height = 30; btn.Width = 150;

btn.Background = Brushes.Transparent;

btn.Content = "버튼";

Stack.Children.Add(btn);

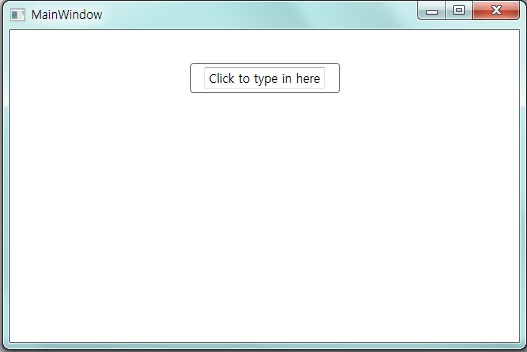
TextBox txt= new TextBox();

txt.Text = "Click to type in here";

btn.Content = txt;

}

}



**예제4) 결과창**

**예제5) StackPanel-Horizontal**

<Window

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

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

Title="Stack Panel" Height="100" Width="400">

<Grid Name="StuffToBuy">

<StackPanel Orientation="Horizontal">

<Image

Source=<http://www.oreilly.com/images/oreilly/add_to_cart.gif> Height="33" Width="142" />

<Button Height="30" Width="150" Background="Transparent">

<TextBox>Click to type in here</TextBox>

</Button>

</StackPanel>

</Grid>

</Window>

XAML 코드를 사용하지 않고 C#코드로만 만들었을 경우

public MainWindow()

{

InitializeComponent();

ex\_2();

}

public void ex\_2()

{

StackPanel Stack = new StackPanel();

Stack.Orientation = Orientation.Horizontal;

this.AddChild(Stack);

//------------

Image img = new Image();

BitmapImage myImg = new BitmapImage();

myImg.BeginInit();

myImg.UriSource = new Uri("http://www.oreilly.com/images/oreilly/add\_to\_cart.gif", UriKind.Absolute);

myImg.DecodePixelWidth = 200;

myImg.EndInit();

img.Source = myImg;

Stack.Children.Add(img);

//----------------

Button btn = new Button();

btn.Height = 30; btn.Width = 150;

btn.Background = Brushes.Transparent;

btn.Content = "버튼";

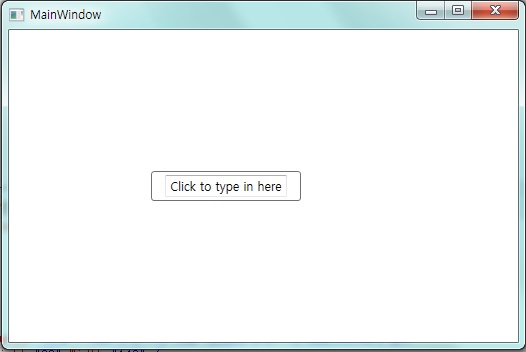
Stack.Children.Add(btn);

TextBox txt = new TextBox();

txt.Text = "Click to type in here";

btn.Content = txt;

}



**예제5-1)결과창**

**예제6) Menu\_Ctrl+x**

메뉴의 항목은 ItemCollectio에 추가할 수 있는 모든 요소가 될 수 있습니다. Menu에서 가장 일반적인 항목 형식은 MenuItem 입니다. MenuItem에서 자식 항목이 포함 될 수 있고, 사용자가 부모 MenuItem을 선택하면 하위 메뉴에 자식 항목이 나타난다.

<Window x:Class="JHY1110.MainWindow"

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

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

Title="Hello and Goodbye" Height="300" Width="300">

<StackPanel Name="HelloWorldGrid" Background="AntiqueWhite">

<Menu Height="21" Margin="0,0,0,0">

<MenuItem Header="Say Hello">

<MenuItem Header="Hello Jesse" InputGestureText="Ctrl+X" />

<MenuItem Header="Hello Alex" />

</MenuItem>

<MenuItem Header="Say Goodbye">

<MenuItem Header="Goodbye Jesse" />

<MenuItem Header="Goodbye Alex" />

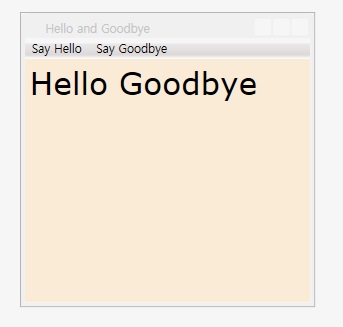
</MenuItem>

</Menu>

<Label Content="Hello Goodbye" FontFamily="Verdana" FontSize="32"/>

</StackPanel>

</Window>



<결과창>

XAML 코드를 사용하지 않고 C#코드로만 만들었을 경우

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

ex\_3();

}

public void ex\_3()

{

StackPanel Stack = new StackPanel();

this.AddChild(Stack);

Menu menu = new Menu();

MenuItem m\_item = new MenuItem();

m\_item.Header = "Say Hello";

MenuItem m\_item1 = new MenuItem();

m\_item1.Header = "Hello Jesse";

MenuItem m\_item2 = new MenuItem();

m\_item2.Header = "Hello Alex";

m\_item2.InputGestureText = "Ctrl+X";

m\_item.Items.Add(m\_item1);

m\_item.Items.Add(m\_item2);

MenuItem m\_item3 = new MenuItem();

m\_item3.Header = "Say Goodbye";

MenuItem m\_item4 = new MenuItem();

m\_item4.Header = "Goodbye Jesse";

MenuItem m\_item5 = new MenuItem();

m\_item5.Header = "Goodbye Alex";

m\_item3.Items.Add(m\_item4);

m\_item3.Items.Add(m\_item5);

menu.Items.Add(m\_item);

menu.Items.Add(m\_item3);

Stack.Children.Add(menu);

}

}

**예제6-1) 메뉴창과 단축키 입력 받았을 때 쓰는 예제소스**

<Window x:Class="JHYqwer.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>

<StackPanel Name="HelloWorldGrid" Background="AntiqueWhite">

<Menu Height="21" Margin="0,0,0,0">

<MenuItem Header="Say Hello" KeyDown="OnKeyDonw">

<MenuItem Header="Hello Jesse" InputGestureText="Ctrl+X" IsCheckable="True" Checked="Jesse\_Checked" Unchecked="Jesse\_UnChecked"/>

<MenuItem Header="Hello Alex" InputGestureText ="Ctrl+Y" IsCheckable="True" Checked="Alex\_Checked" Unchecked="Alex\_UnChecked" />

</MenuItem>

<MenuItem Header="Say Goodbye">

<MenuItem Header="Goodbye Jesse" />

<MenuItem Header="Goodbye Alex" />

</MenuItem>

</Menu>

<Label Name="label1" Content="Hello Goodbye" FontFamily="Verdana" FontSize="32" />

</StackPanel>

</Grid>

</Window>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

public void OnKeyDonw(object sender, KeyEventArgs e)

{

if (Keyboard.IsKeyDown(Key.LeftCtrl))

{

if (e.Key == Key.X)

{

label1.Content = "Hello Jesse";

}

}

else

{

label1.Content = "Hello Goodbye";

}

if (Keyboard.IsKeyDown(Key.LeftCtrl))

{

if (e.Key == Key.Y)

{

label1.Content = "Hello Alex";

}

}

else

{

label1.Content = "Hello Goodbye";

}

}

private void Jesse\_Checked(object sender, RoutedEventArgs e)

{

label1.Content = "Hello Jesse";

}

private void Alex\_Checked(object sender, RoutedEventArgs e)

{

label1.Content ="Hello Alex";

}

private void Jesse\_UnChecked(object sender, RoutedEventArgs e)

{

label1.Content ="Hello Goodbye";

}

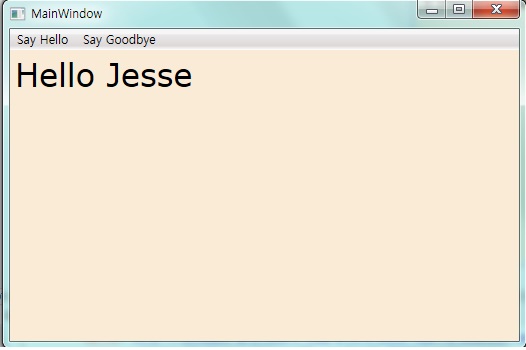
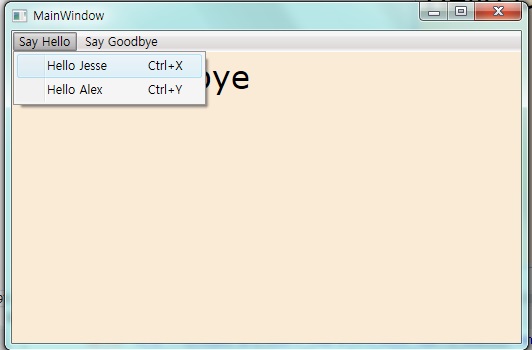
private void Alex\_UnChecked(object sender, RoutedEventArgs e)

{

label1.Content ="Hello Goodbye";

}

}



**6\_1)예제 결과창**