codingOn

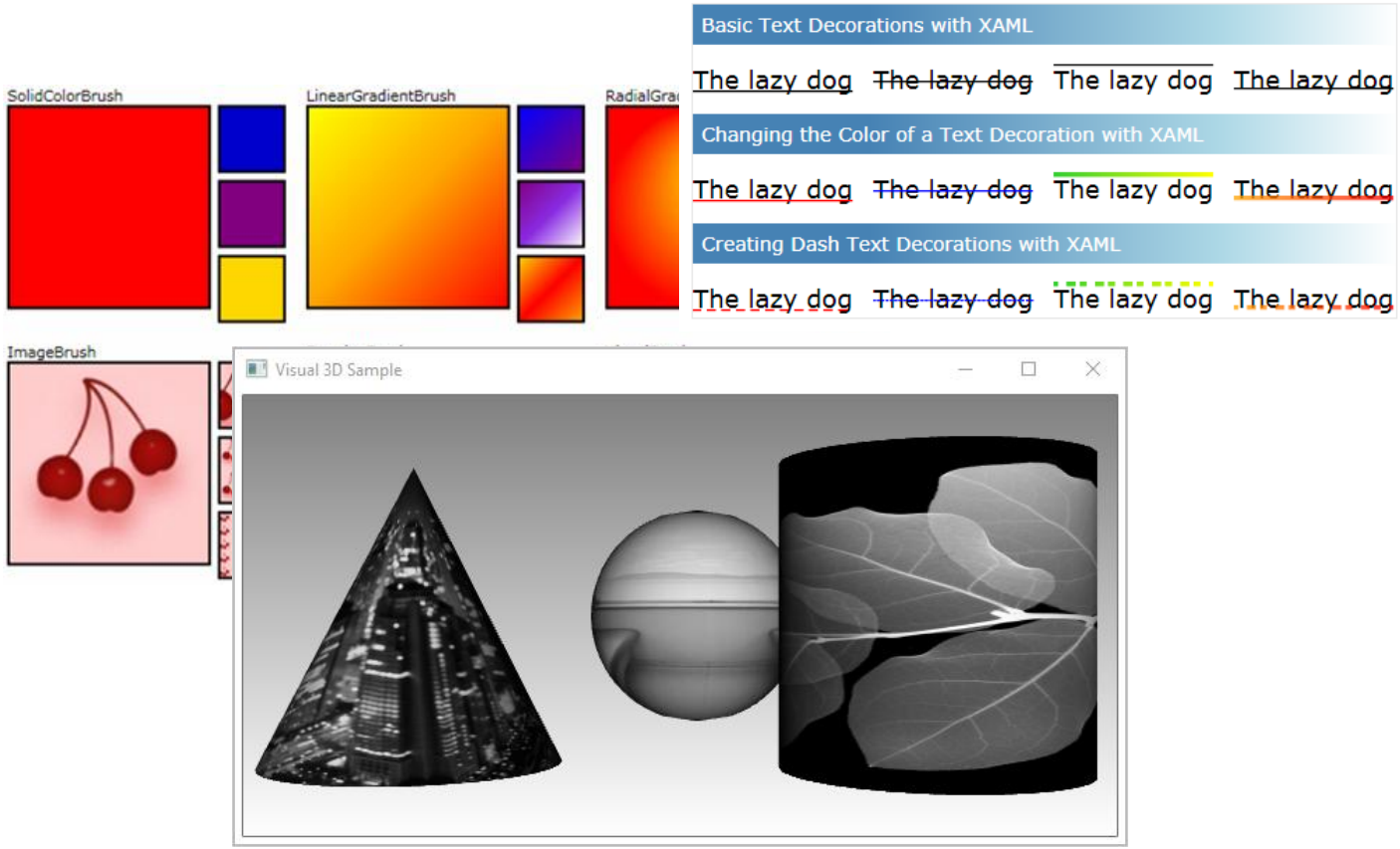
C# WPF

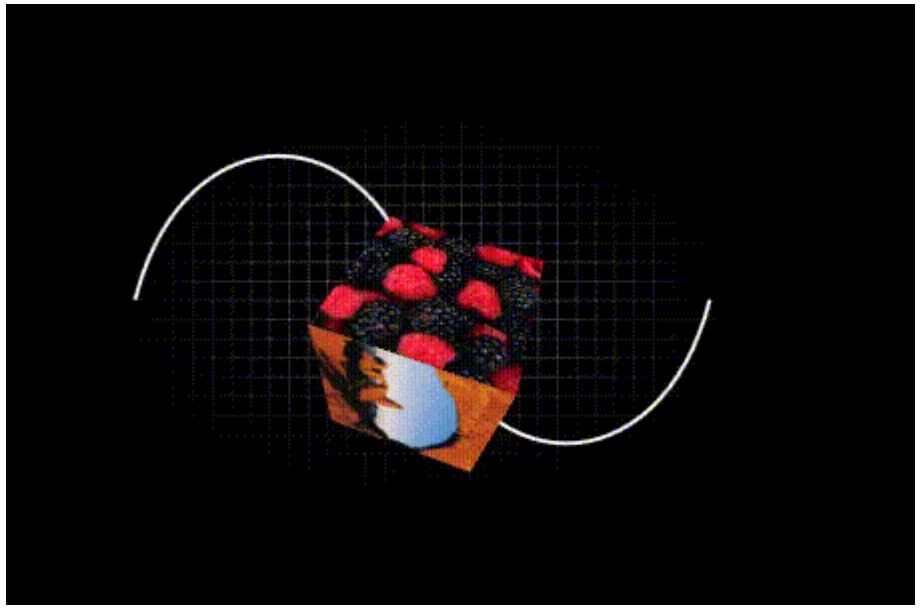
WPF란?

- Windows Presentation Foundation의 약자
- Microsoft에서 2006년에 만든 UI 프레임워크
- .NET 환경에서 동작, 주로 C#을 사용
  - .NET으로 빌드가 가능 = Visual Studio 에서 작업해야 함
- **벡터 기반 렌더링** (해상도 변경에 따른 화질 저하가 없음)
  - ***벡터 기반(Vector) = 수학적 좌표로 도형을 그림.**
  - ***렌더링 = 컴퓨터가 내부 데이터를 화면에 그리는 작업**

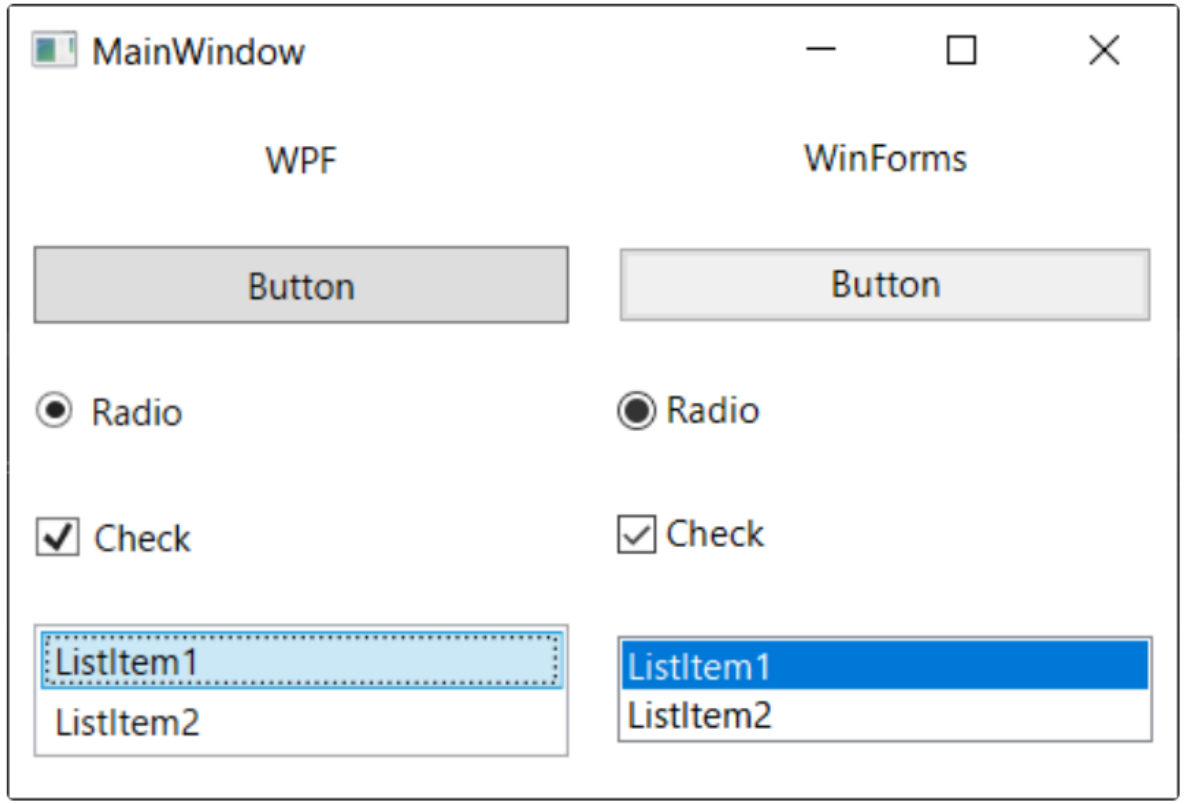
WPF 그래픽 표현

https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview/?view=netdesktop-7.0





WPF vs WinForm



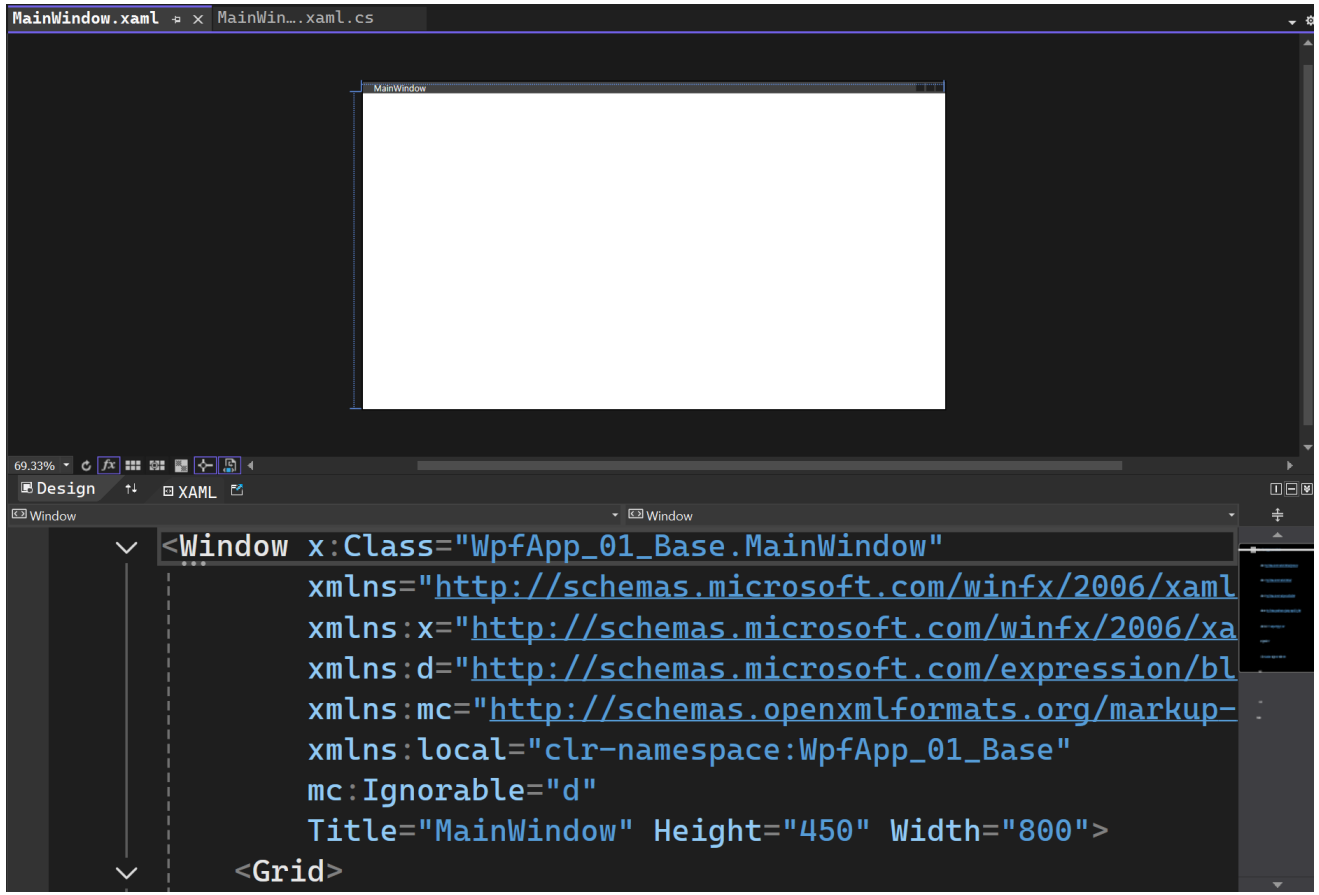
WPF (왼쪽)

- 벡터 형식 UI
- 개발 시간이 오래 걸림
- 표준 컨트롤에 의존하지 않기 때문에 비교적 자유롭게 커스터마이징 가능
- UI를 제작하는데 XAML 코드를 사용
- WinForm 보다 사용자 측면(UI)이 강조됨

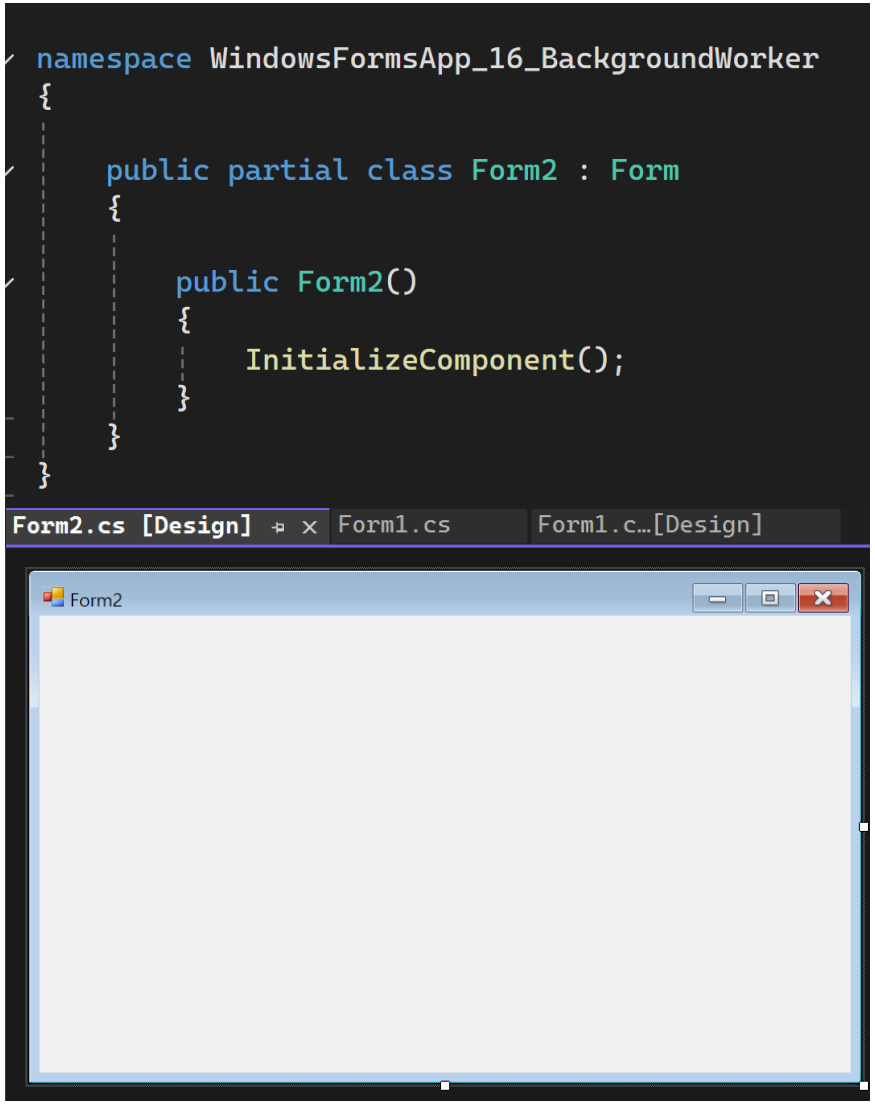
WinForm (오른쪽)

- 디자이너 툴에 의존하기 때문에 개발이 쉬움
- 덕분에 개발 시간이 짧음
- 하지만, 확장성이 떨어짐
- Visual Studio 디자인 툴을 사용

WPF vs WinForm



WPF



WinForm

XAML

- eXtensible Application Markup Language (HTML, XML 등)
- WPF에서 View를 그리기위한 언어
- Markup Language 이기 때문에 태그를 사용

```
<Style TargetType="{x:Type Button}"
       BasedOn="{x:Null}">
    <Setter Property="Background"
            Value="{DynamicResource NormalBrush}" />
    <Setter Property="Foreground"
            Value="{DynamicResource TextBrush}" />
    <Setter Property="BorderBrush"
            Value="{DynamicResource NormalBorderBrush}" />
    <Setter Property="Template"
            Value="{DynamicResource ButtonTemplate}" />
    <Setter Property="FontSize"
            Value="14" />
</Style>
```

태그 열기

다른 태그를 감싸고 있는 태그가 부모(Style), 감싸져 있는 태그가 자식(Setter) 태그

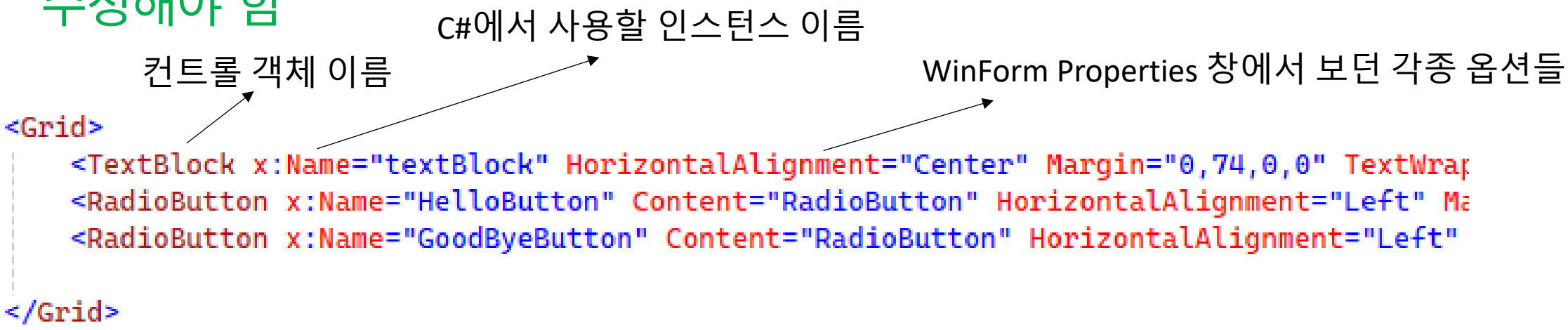
Property, Value 같이 설정 가능한 요소들을 속성이라고 부름

태그 열고, 닫기 한 줄로 표형

태그 닫기

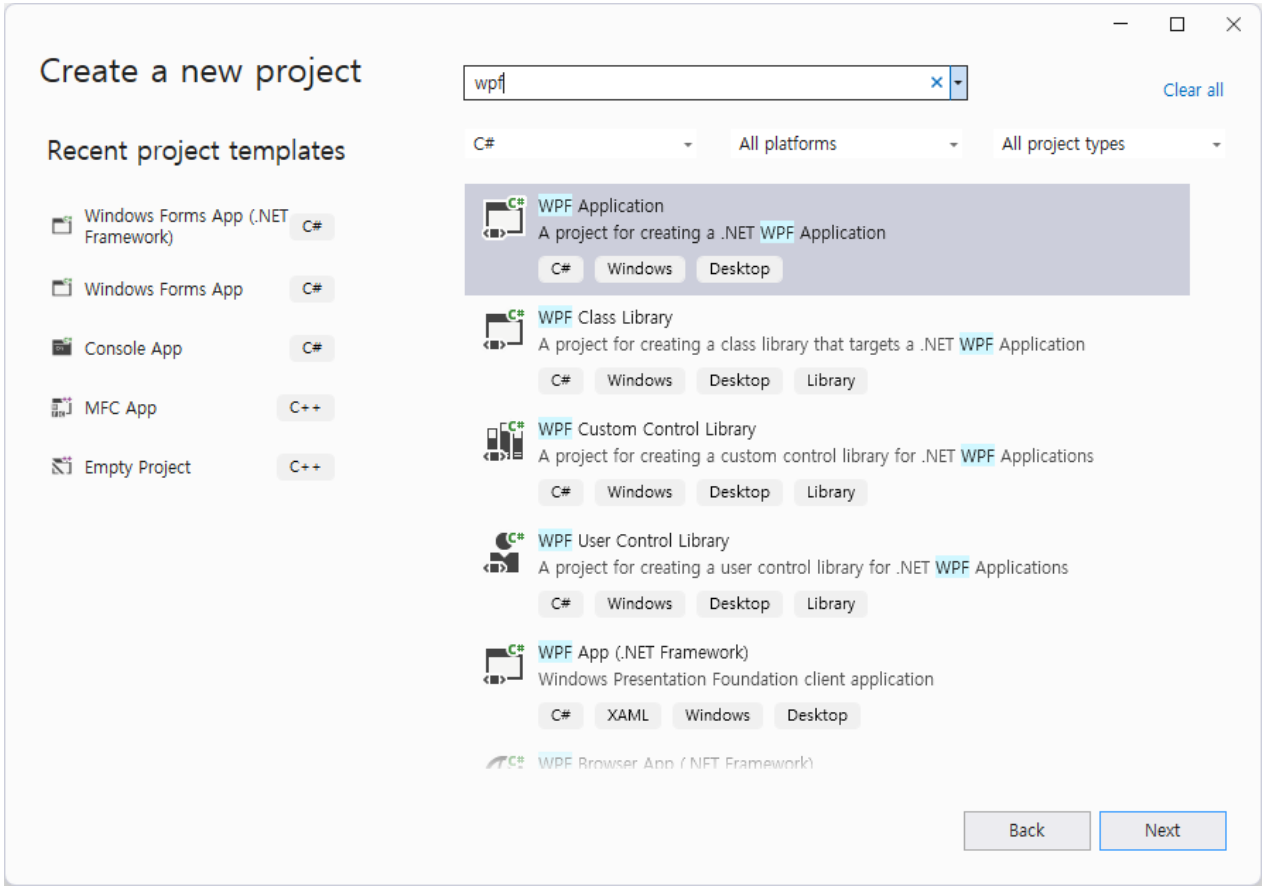
XAML

- 대표적인 Markup Language인 HTML과 비교
  - HTML : 정해진 태그 (head, body, h1, div 등)만 사용 가능
  - XAML : Microsoft에서 개발한 XML, UI와 데이터를 주고 받는 목적
- Visual Studio에서 자동 생성해주기도 하지만, 설정값이나 옵션은 직접 수정해야 함



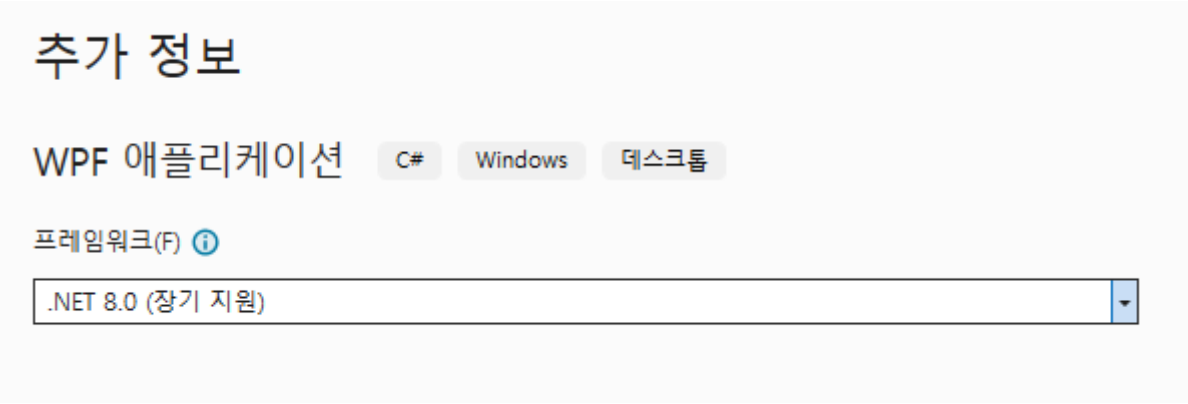
WPF 프로젝트 생성

- Visual Studio 프로젝트 생성에서 WPF Application을 선택

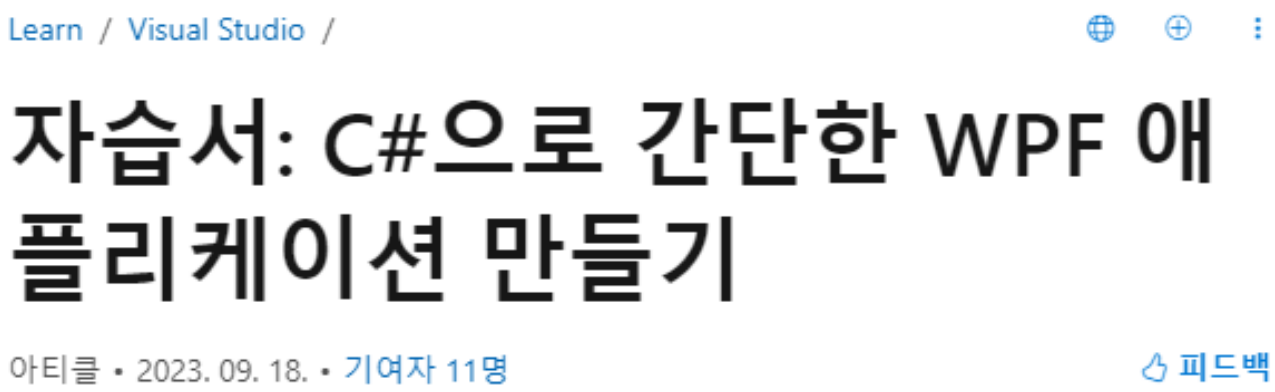


WPF 프로젝트 생성

- WPF 프로젝트는 .NET 최신버전을 사용 가능
- WinForm은 아직 사용되고는 있지만 .NET Framework 4.8 이후로는 업데이트가 중단됨



실습. WPF 공식 자습서-1



https://learn.microsoft.com/ko-kr/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2022

실습. WPF 공식 자습서-2

Learn / .NET / Windows Presentation Foundation / C#

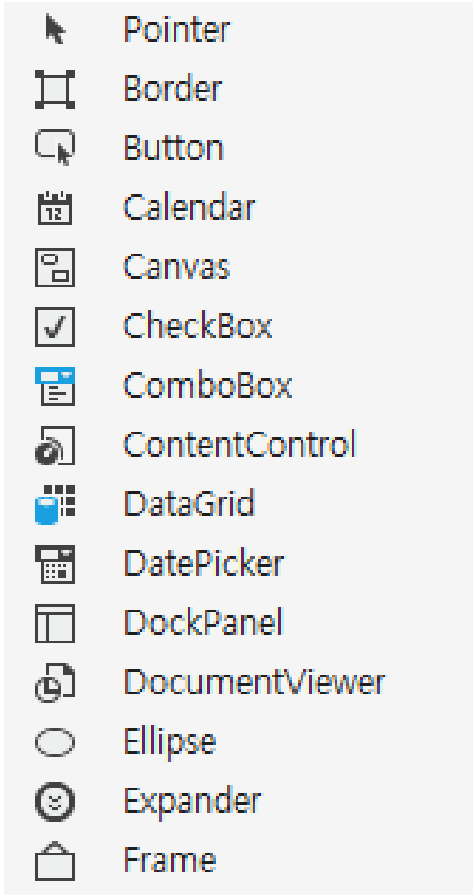
자습서: Visual Studio 2019에서 첫 번째 WPF 애플리케이션 만들기

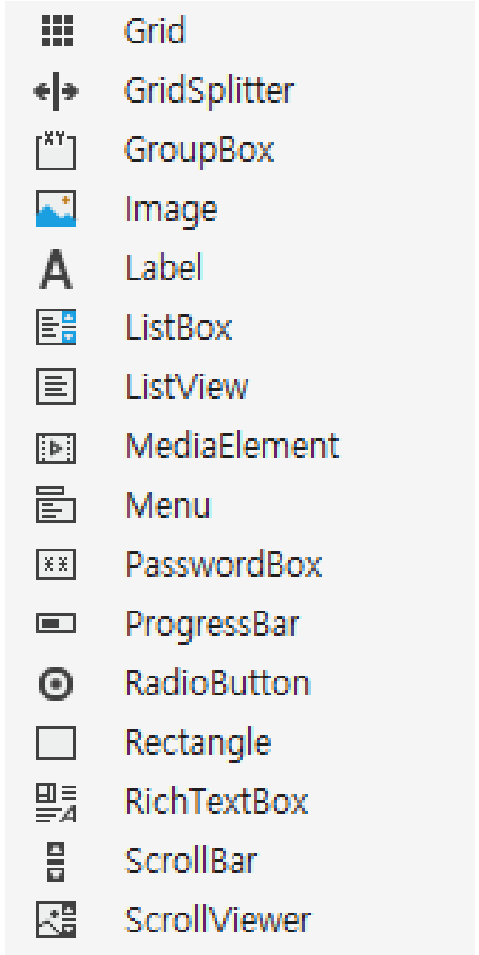
아티클 • 2024. 12. 19. • 기여자 4명 피드백

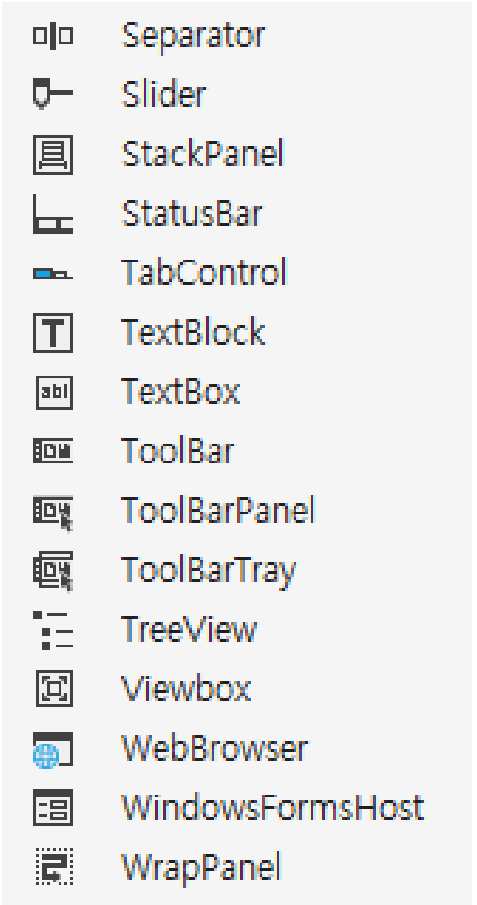
https://learn.microsoft.com/ko-kr/dotnet/desktop/wpf/getting-started/walkthrough-my-first-wpf-desktop-application?view=netframeworkdesktop-4.8

WPF 도구상자

WPF 도구상자

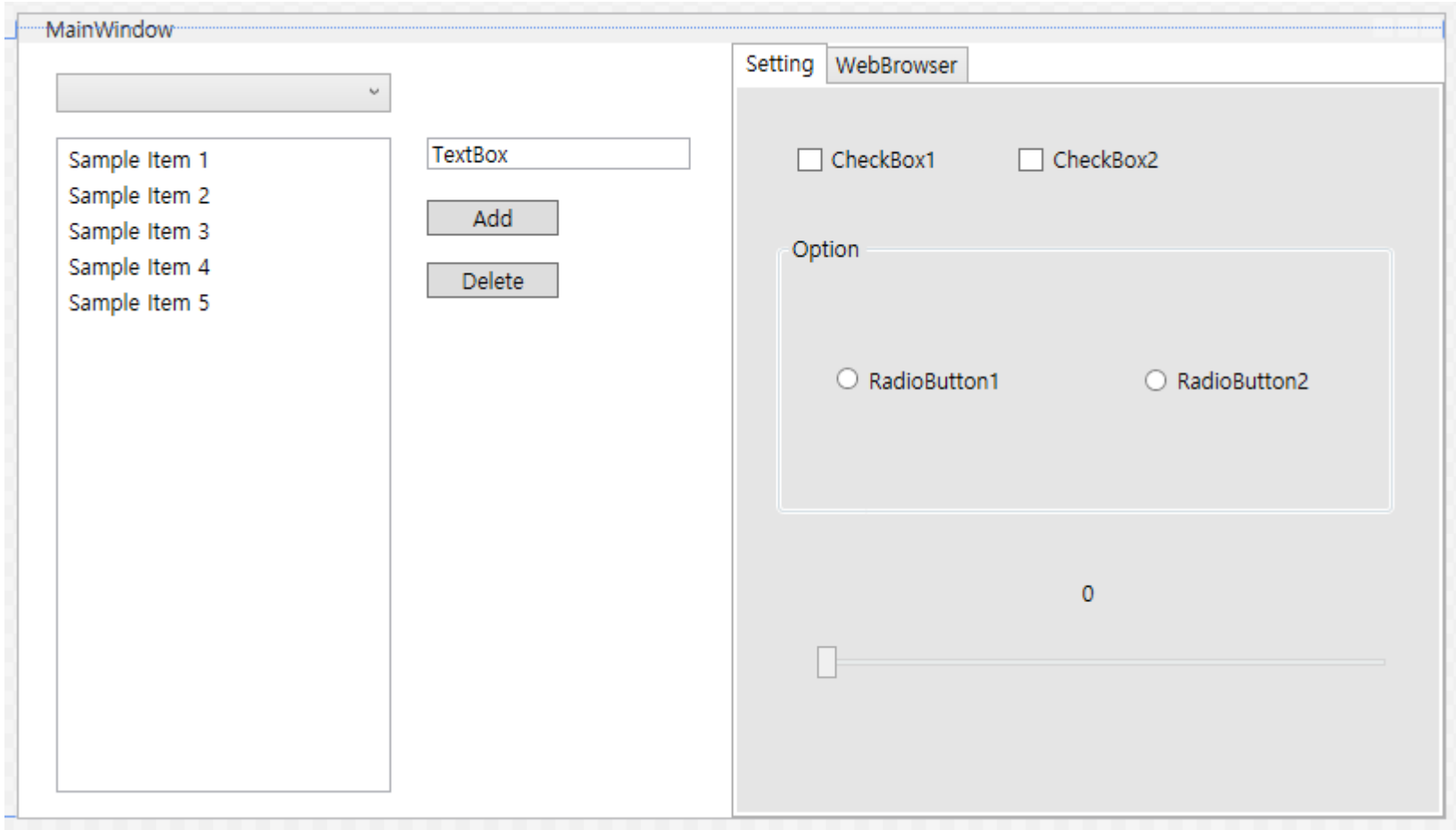

Pointer
Border
Button
Calendar
Canvas
CheckBox
ComboBox
ContentControl
DataGrid
DatePicker
DockPanel
DocumentViewer
Ellipse
Expander
Frame


Grid
GridSplitter
GroupBox
Image
Label
ListBox
ListView
MediaElement
Menu
PasswordBox
ProgressBar
RadioButton
Rectangle
RichTextBox
ScrollBar
ScrollViewer


Separator
Slider
StackPanel
StatusBar
TabControl
TextBlock
TextBox
ToolBar
ToolBarPanel
ToolBarTray
TreeView
Viewbox
WebBrowser
WindowsFormsHost
WrapPanel

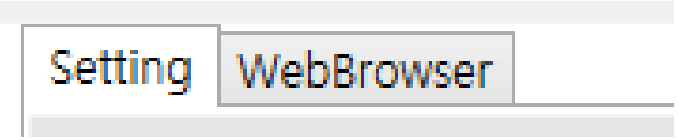
도구상자 - 기본 기능 종합

- 탭 컨트롤
- 그룹 박스
- 라디오 버튼
- 체크 박스
- 콤보 박스
- 리스트 박스
- 슬라이더
- 웹 브라우저



도구상자 - 탭 컨트롤

- 각종 컨트롤을 탭에 포함시켜 한정된 공간에 여러개의 컨트롤을 넣을 수 있음
- TabItem 태그를 사용하여 탭의 수를 늘릴 수 있음
- 탭을 변경시 탭에 포함된 컨트롤이 모두 표시/숨김 처리됨




```
<TabControl x:Name="tabControl1" Margin="400,0,0,0">
    <TabItem Header="Setting">
        <Grid Background="#FFE5E5E5">
            <CheckBox x:Name="checkBox1" Content="CheckBox1" HorizontalAlignment=
            <CheckBox x:Name="checkBox2" Content="CheckBox2" HorizontalAlignment=
            <GroupBox x:Name="Options1" Header="Option" Margin="22,81,25,167">
                <Grid>
                    <RadioButton x:Name="radioButton1" Content="RadioButton1" Hor
                    <RadioButton x:Name="radioButton2" Content="RadioButton2" Hor
                </Grid>
            </GroupBox>
            <Slider x:Name="slider1" HorizontalAlignment="Left" Margin="45,313,0,
            <Label x:Name="label_sliderValue" Content="0" HorizontalAlignment="Ce
        </Grid>
    </TabItem>
    <TabItem Header="WebBrowser">
        <Grid Background="#FFE5E5E5">
            <WebBrowser x:Name="WebBrowser1"/>
        </Grid>
    </TabItem>
</TabControl>
```

* 컨트롤의 위치를 자유롭게 변경하기 위해서는
Grid 안에 컨트롤을 생성하는 것이 편함

도구상자 – 레이아웃 패널

- WPF에서 요소를 배치 할 때, 반드시 레이아웃 패널로 감싸야 한다.
- Why?
  - 모든 UI 요소가 “계층(Tree) 구조”로 구성되어 있고,
  - 자식 컨트롤들이 어디에, 어떻게 배치될지 결정하는 주체가 필요.
  - 그 역할이 레이아웃 패널.

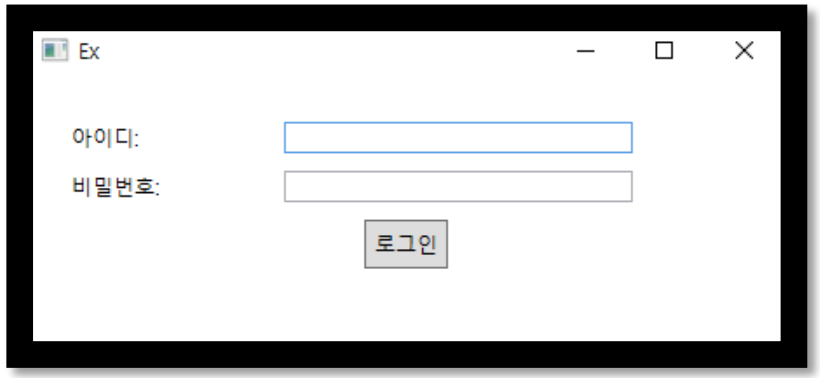
- 종류
  - Grid
  - StackPanel
  - Canvas

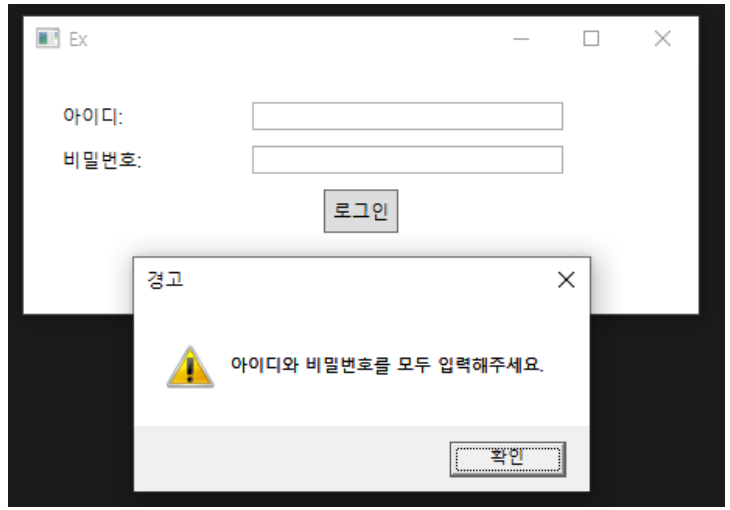
도구상자 – 레이아웃 패널

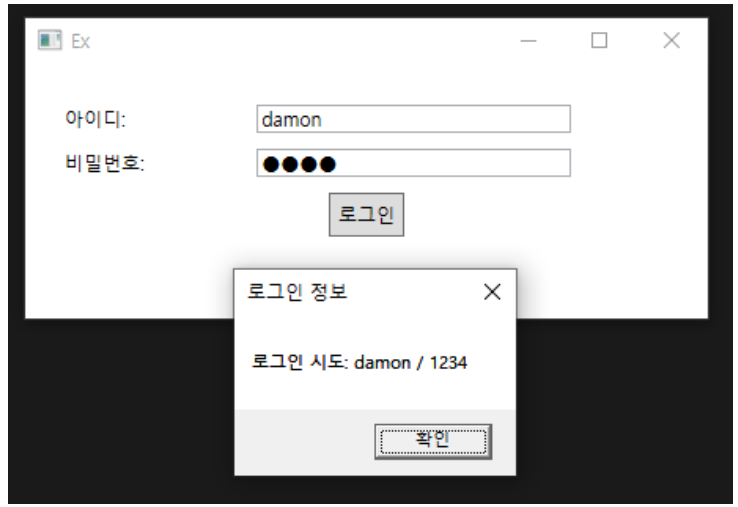
- Grid
  - 행과 열 기반 정렬
  - 정렬 + 반응형에 최적화
- StackPanel
  - 위아래/가로 순차 정렬
  - 간단한 정렬
- Canvas
  - 절대 위치 좌표 정렬
  - 자유 배치 (ex. 게임, 편집 etc...)
  - 실무에서 잘 쓰이지 않음.

실습. Grid – 로그인 폼 만들기

- Gird를 이용하여 결과화면 처럼 만들기.
- “로그인” 버튼은 (열 합치기) 사용.
- 이벤트 메서드 생성
  - ID/PW 입력 받고 로그인 버튼 클릭
  - 입력받은 ID/PW 화면에 메시지 출력.

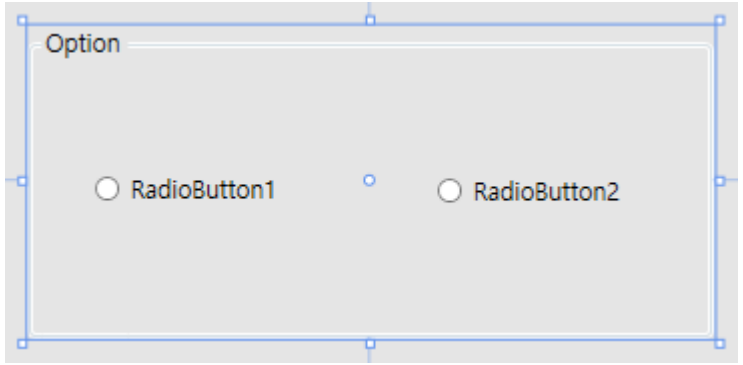






도구상자 - 그룹 박스

- 비슷한 역할을 하는 컨트롤을 그룹으로 묶어 정리하는 용도
- 기능보다는 시각적으로 정돈된 느낌을 주기 위함

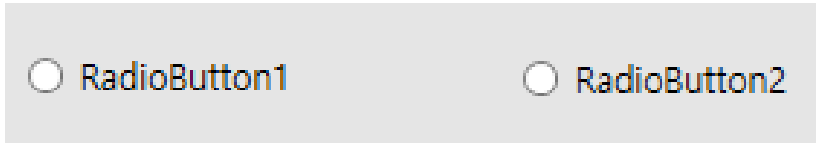



```
<GroupBox x:Name="Options1" Header="Option" Margin="22,81,25,167">
    <Grid>
        <RadioButton x:Name="radioButton1" Content="RadioButton1" Horiz
        <RadioButton x:Name="radioButton2" Content="RadioButton2" Horiz
    </Grid>
</GroupBox>
```

* 컨트롤의 위치를 자유롭게 변경하기 위해서는
Grid 안에 컨트롤을 생성하는 것이 편함

도구상자 - 라디오 버튼

- 사용자의 중복된 선택을 막고 하나의 선택을 유도하기 위한 컨트롤
- IsChecked 속성으로 체크 여부를 결정
- Checked/Unchecked 속성에 C#으로 제어할 이벤트 핸들러(메소드)를 지정
  - 더블 클릭시 메소드가 자동으로 생성됨
- GroupName을 설정하여 한 그룹 내에서 하나의 옵션만 선택할 수 있게 가능




```
<RadioButton x:Name="radioButton1"
             Content="RadioButton1"
             IsChecked="True"
             Checked="radioButton1_Checked"
```

도구상자 - 라디오 버튼

- C#에서 Checked 이벤트 핸들러 메소드를 사용 가능

```
// 라디오 버튼이 체크될 때 실행됨, 체크 해제 시 실행 안됨
1 reference
private void radioButton1_Checked(object sender, RoutedEventArgs e)
{
    if(radioButton1.IsChecked == true)
    {
        // 라디오 버튼이 체크되었을 경우 실행할 코드
    }
}
```

도구상자 - 슬라이더

- 특정한 값을 실시간으로 변화시키기 위해 사용
- Tick 관련 속성으로 이동 단위를 임의로 설정 가능
- C#에서 ValueChanged 이벤트로 값 제어 가능

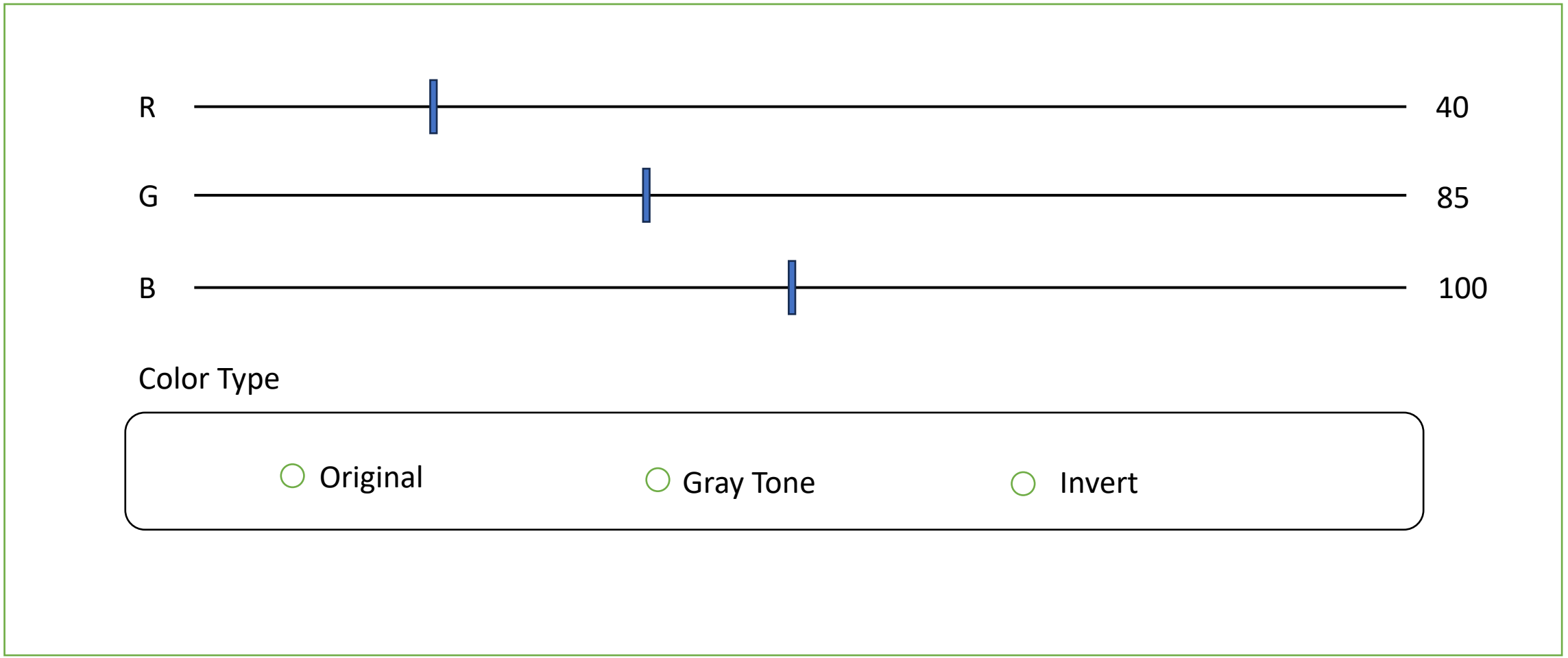

```
<Slider x:Name="slider1"
        Maximum="100"
        TickPlacement="BottomRight"
        TickFrequency="5"
        IsSnapToTickEnabled="True"
```

```
1 reference
private void slider1_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)
{
    label_sliderValue.Content = slider1.Value.ToString();
}
```

실습. 컬러 슬라이더

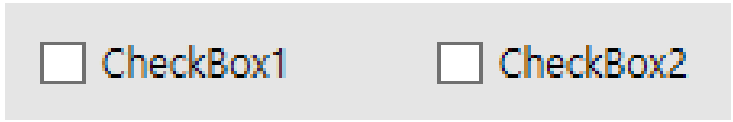
- 슬라이더 3개를 넣고 각각 Label로 Red, Green, Blue 이름을 붙임
- 범위는 0~255를 가지고, 5 단위로 움직임
- 각 슬라이더의 현재 값을 Label 또는 TextBox로 표현
- 값을 변경할 때마다 지정된 값 대로 Grid의 배경색 변경
  - Color.FromRgb() 메서드 검색
  - SolidColorBrush() 검색
- 그룹 박스 안에 일반, 반전, 흑백 라디오 버튼을 각각 만들고 해당 기능에 따라 배경 색상이 바뀌도록
  - 흑백은 RGB 값을 **평균**내서 모든 컬러 채널에 일괄 적용
  - 반전은 RGB 각각의 컬러 채널에서 최대값인 255에서 현재 값을 **뺀 값**을 적용

실습. 컬러 슬라이더



도구상자 - 체크 박스

- 사용자에게 중복 선택을 허용할 때 사용
- IsChecked 속성으로 체크 여부 설정
- Checked/Unchecked 속성에 이벤트 핸들러로 메소드를 등록하여 C#에서 제어
  - 더블 클릭시 자동으로 생성




```
<CheckBox x:Name="checkBox1"
          IsChecked="False"
          Checked="checkBox1_Checked"
          Content="CheckBox1"
```

```
// 체크 박스가 체크되면 실행됨, 체크 해제 시 실행 안됨
1 reference
private void checkBox1_Checked(object sender, RoutedEventArgs e)
{
    if(checkBox1.IsChecked == true)
    {
        // 체크 박스가 체크 되었을 경우 실행될 코드
    }
}
```

도구상자 - 콤보 박스

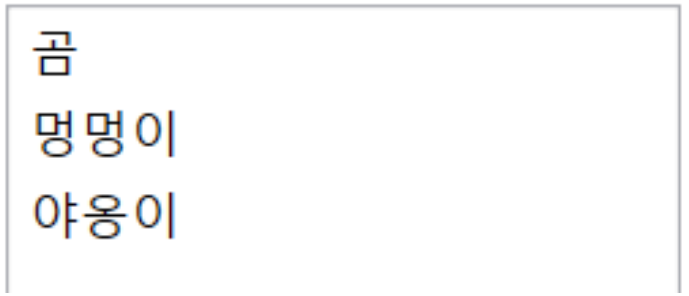
- 여러가지 옵션 중 하나를 선택하고 나머지 옵션을 숨길 때 사용
- ComboBoxItem 태그를 사용하여 요소 추가
- C#에서 요소를 추가, 제거, 선택된 아이템 정보 가져오기를 할 수 있음

```
<ComboBox x:Name="comboBox" HorizontalAlignment=
    <ComboBoxItem Content="Coffee"/>
    <ComboBoxItem Content="Milk"/>
    <ComboBoxItem Content="Hot Choco"/>
</ComboBox>
```

```
comboBox.Items.Add("Tomato Juice");
ComboBoxItem item = (ComboBoxItem)comboBox.SelectedValue;
string item_name = item.Content.ToString();
comboBox.Items.Remove(comboBox.SelectedItem);
```

도구상자 - 리스트 박스

- 데이터를 리스트 형태로 나타냄
- C#에서 리스트의 각 요소들을 편집 가능

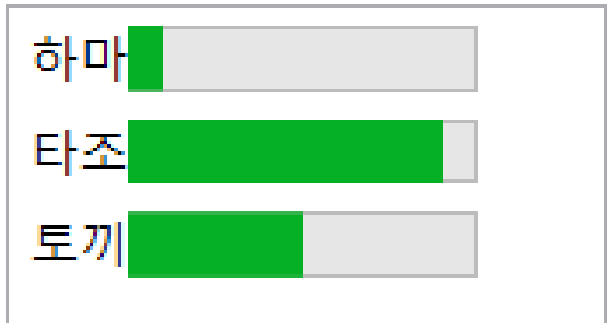



```
<ListBox x:Name="listBox" Margin="21,53,591,14">
    <ListBoxItem>곰</ListBoxItem>
    <ListBoxItem>멍멍이</ListBoxItem>
    <ListBoxItem>야옹이</ListBoxItem>
</ListBox>

listBox.Items.Add("악어");
listBox.Items.Add("병아리");
listBox.Items.Add("펭귄");
```

도구상자 - 리스트 박스 응용 - 데이터 바인딩

- 리스트 박스를 포함한 대부분의 컨트롤들은 데이터 템플릿에 데이터를 바인딩(Binding)하여 사용 가능

```
<ListBox x:Name="listBox" Margin="21,53,591,14">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <Grid Margin="0,2">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="*"/>
                    <ColumnDefinition Width="100"/>
                </Grid.ColumnDefinitions>
                <TextBlock Text="{Binding Name}"/>
                <ProgressBar Grid.Column="1" Minimum="0" Maximum="100" Value="{Binding Percent}"/>
            </Grid>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```



도구상자 - 리스트 박스 응용 - 데이터 바인딩

- C#에서 바인딩할 데이터 변수를 가진 클래스를 작성
- 해당 클래스로 리스트를 만들어서 리스트 박스에 바인딩하는 방식

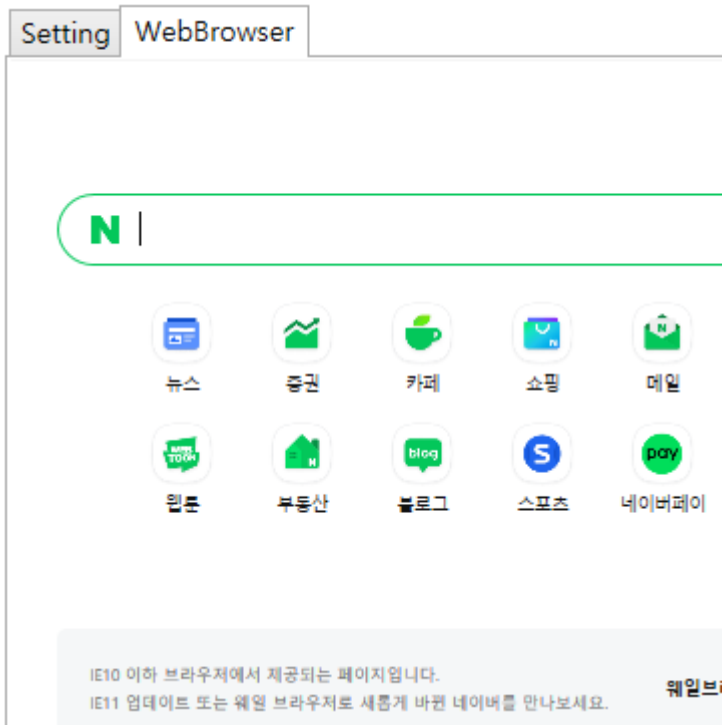
```
5 references
class Animals
{
    3 references
    public string Name { get; set; }
    3 references
    public int Percent { get; set; }
}
```

```
List<Animals> animals= new List<Animals>();
animals.Add(new Animals() { Name = "하마", Percent = 10 });
animals.Add(new Animals() { Name = "타조", Percent = 90 });
animals.Add(new Animals() { Name = "토끼", Percent = 50 });

listBox.ItemsSource = animals;
```

도구상자 - 웹 브라우저(IE)

- 일반 웹 브라우저와 기능은 동일
- C#에서 Navigate 메소드를 사용하여 원하는 페이지로 이동 가능

```
<WebBrowser x:Name="WebBrowser1"/>

WebBrowser1.Navigate("http://www.naver.com");
WebBrowser1.GoBack(); // 뒤로가기
WebBrowser1.GoForward(); // 앞으로가기
```

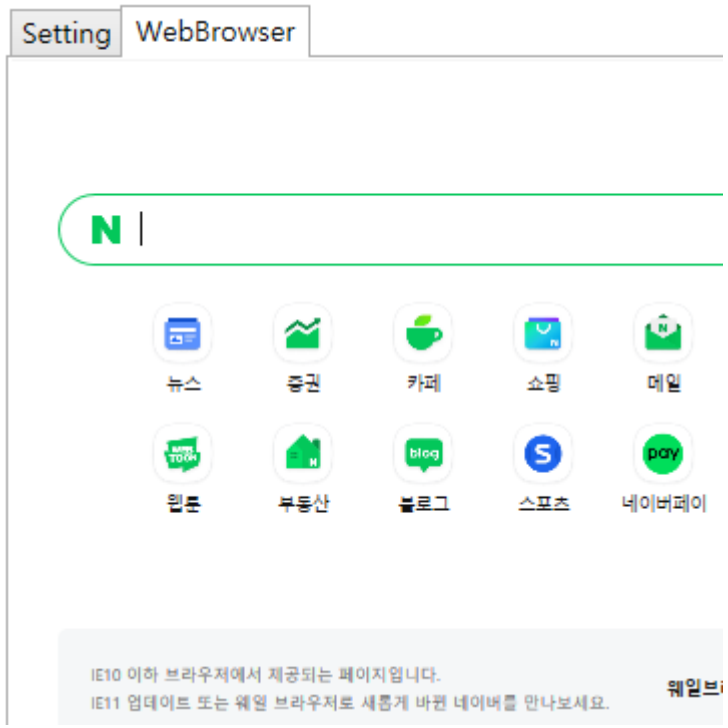


도구상자 - 웹 브라우저(Edge)

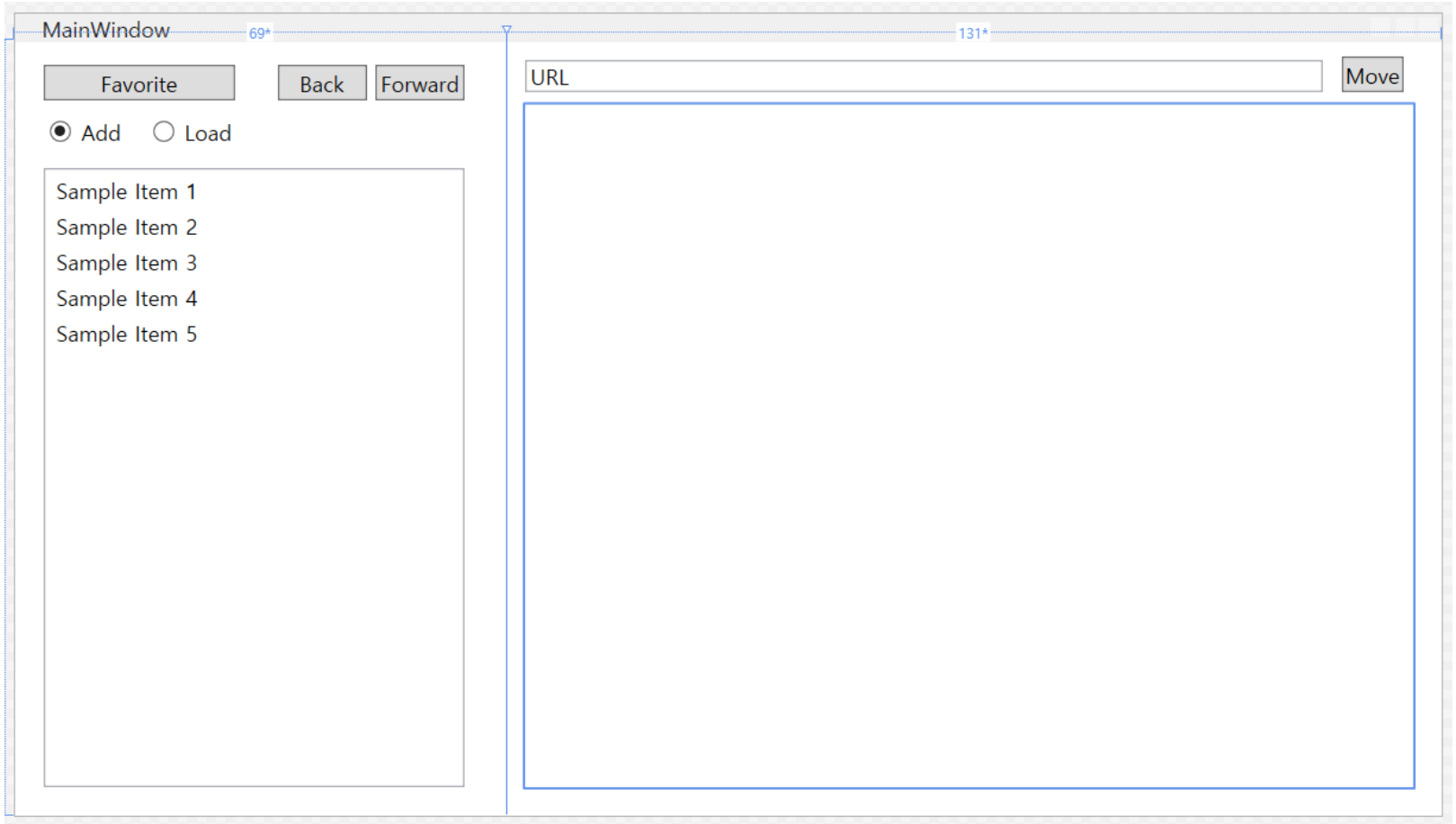
- 설치방법
  - https://learn.microsoft.com/en-us/microsoft-edge/webview2/get-started/wpf

```
<DockPanel>
  <wv2:WebView2 Name="webView"
                Source="https://www.microsoft.com"
  />
</DockPanel>
```

```
webView.CoreWebView2.Navigate(addressBar.Text);
```



실습. 심플한 웹 브라우저


MainWindow
69*
131*
Favorite
Back
Forward
URL
Move
Add
Load
Sample Item 1
Sample Item 2
Sample Item 3
Sample Item 4
Sample Item 5

실습. 심플한 웹 브라우저

1. 아래 컨트롤을 사용하여 웹 브라우저 구성
   - 리스트 박스, 웹 브라우저, 텍스트 박스(URL),
   - 페이지 이동 버튼, 뒤로 가기 버튼, 앞으로 가기 버튼,
   - 즐겨찾기 버튼, 라디오 버튼 저장, 라디오 버튼 불러오기
2. 페이지 이동 - URL에 적힌 주소로 페이지 이동
3. 뒤로 가기, 앞으로 가기 - 버튼 클릭 시 기존 웹 브라우저와 동일한 기능 수행

실습. 심플한 웹 브라우저

4. 즐겨찾기
   - 라디오 버튼이 Add로 돼있으면 현재 URL을 리스트 박스에 추가
   - 라디오 버튼이 Load로 돼있으면 리스트 박스에 선택된 URL로 웹 브라우저가 이동
5. 파일 사용
   - 리스트 박스에 변동이 있을때마다 .fvr 파일에 리스트 요소를 저장
   - 프로그램이 켜질 때 .fvr 파일을 불러와서 리스트 박스의 내용을 채움