

[C# 기반]

WPF GUI Programming 심화

송 영 욱

Advanced WPF Programming

Contents

- WPF 기본 요약
- WPF RESOURCE
- WPF TRIGGER
- WPF COMMAND
- WPF 파일 연동
- Database 연동
- 데이터 시각화
- 프로젝트

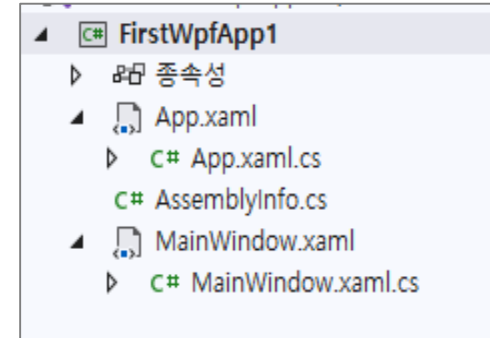
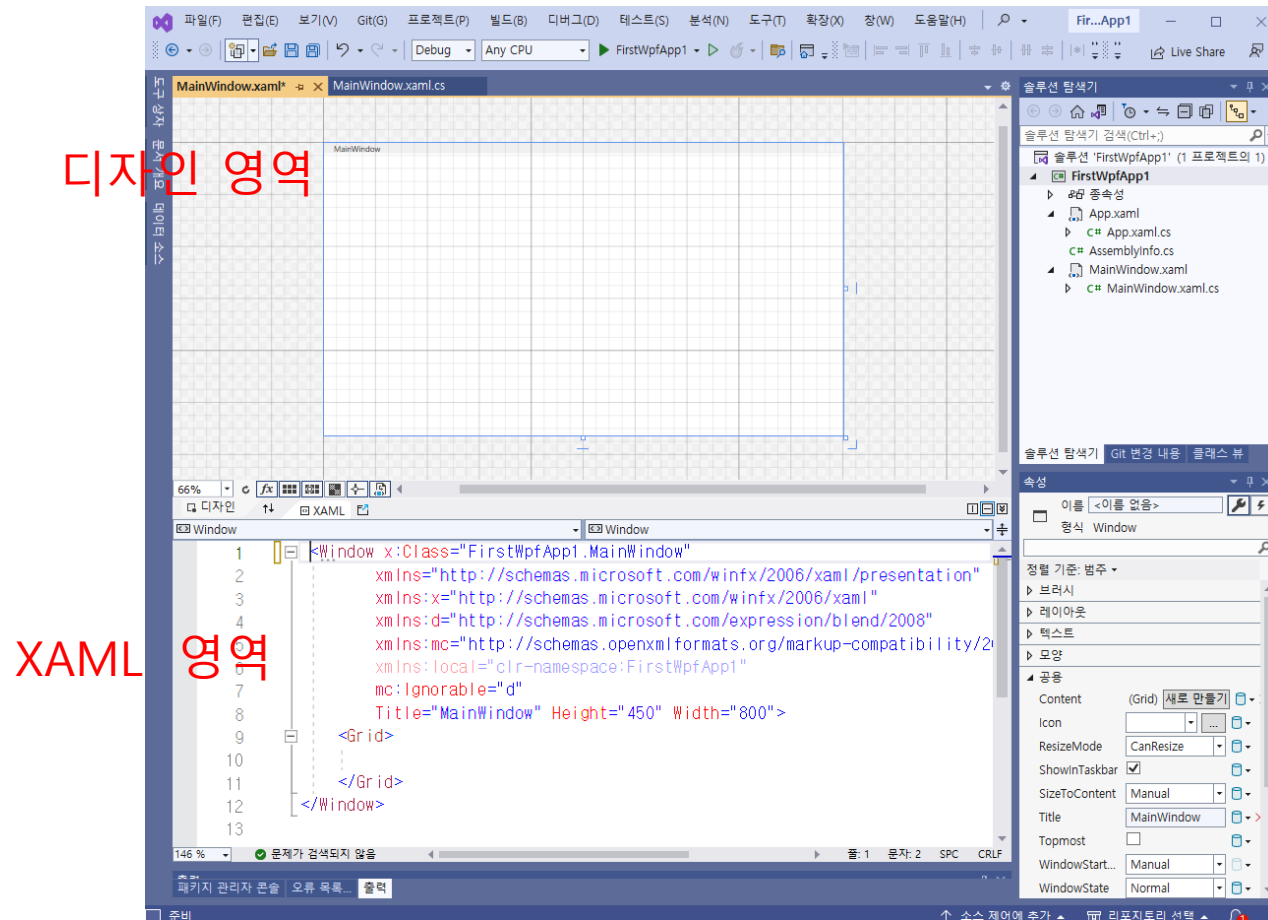
[C# 기반]

Advanced WPF Programming

WPF 기본 요약

Advanced WPF Programming

WPF 프로젝트 구조 이해



.cs
.xaml 파일로 구성

Advanced WPF Programming

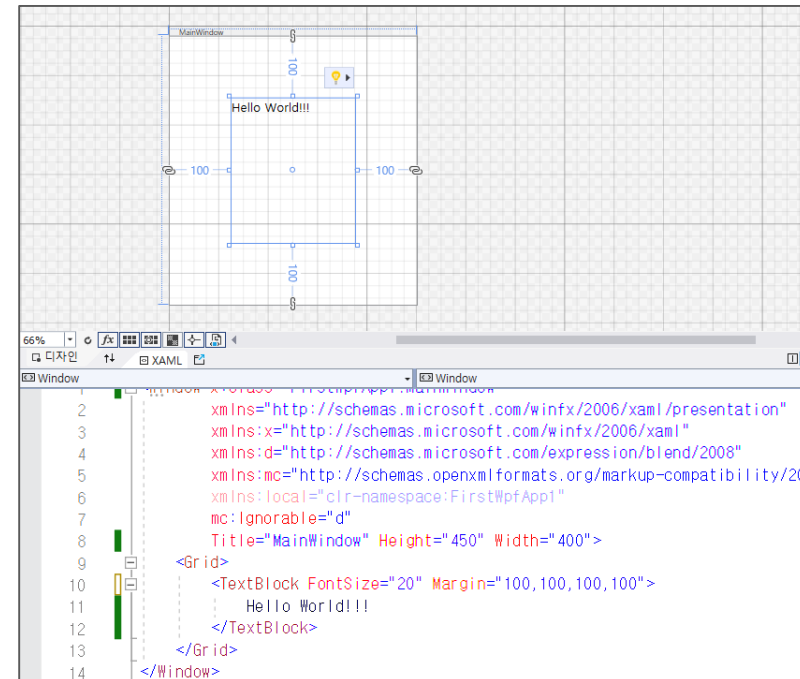
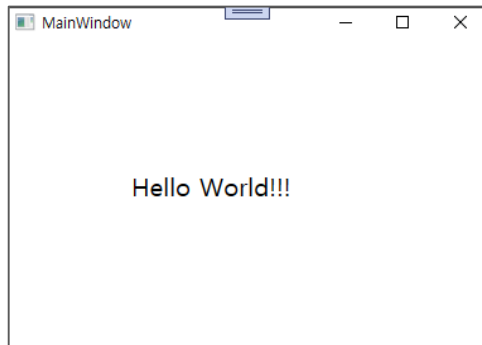
WPF 프로젝트 구조 이해

- App.xaml

```
1 <Application x:Class="FirstWpfApp1.App"
2     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4     xmlns:local="clr-namespace:FirstWpfApp1"
5     StartupUri="MainWindow.xaml">
6     <Application.Resources>
7
8     </Application.Resources>
9 </Application>
```

- MainWindow.xaml 코드 및 디자인

- 실행결과



Advanced WPF Programming

WPF 프로젝트 구조 이해

- App.xaml의 StartupUri 종류

유형	시간 범위	애플리케이션 종류
Window	Window	독립 실행형 전용
NavigationWindow	NavigationWindow	독립 실행형 전용
Page	NavigationWindow	독립 실행형/브라우저 호스팅
UserControl	NavigationWindow	독립 실행형/브라우저 호스팅
FlowDocument	NavigationWindow	독립 실행형/브라우저 호스팅
PageFunction<T>	NavigationWindow	독립 실행형/브라우저 호스팅

Advanced WPF Programming

WPF 프로젝트 구조 이해

- App.xaml의 StartupUri 및 Startup 이해
 - WPF App Project를 생성하면 App.xaml과 MainWindow.xaml 이 기본으로 생성된다.
 - App 클래스는 Application을 상속받은 것으로 프로그램의 시작과 종료를 담당한다.
 - MainWindow는 화면에 보이는 메인 UI다.
- StartupUri 속성
 - App.xaml의 StartupUri 속성에 MainWindow.xaml이 명시되어 있기 때문에 프로그램 실행시 메인 UI가 보이게 된다.

```
<Application x:Class="FirstWPF.App"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="clr-namespace:FirstWPF"
  StartupUri="MainWindow.xaml">
```

Advanced WPF Programming

WPF 프로젝트 구조 이해

– Startup 이벤트

- StartupUri 속성 대신에 Startup 이벤트 핸들러를 통해 MainWindow를 보이게 할 수도 있다.

```
<Application x:Class= " FirstWPF.App"  
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  
  xmlns:local="clr-namespace: FirstWPF"  
  Startup="Application_Startup">
```

- App.cs에 Application_Startup 메소드 구현

```
private void Application_Startup(object sender, StartupEventArgs e)  
{  
    var mainWindow = new MainWindow();  
    mainWindow.Show();  
}
```


Advanced WPF Programming

WPF 프로젝트 구조 이해

- App.xaml의 OnStartup 오버라이딩 이해
 - 처음 실행되는 윈도우에서 Login 정보를 확인한 후 인증을 거친 경우에만 작업한 MainWindow를 열 때 이용할 수 있는 방법
 - LoginWindow의 인증이 거치면 LoginWindow는 종료되고 MainWindow가 열려도록 설정
 - StartUri 또는 StratUp을 이용하지 않는다.
 - App.xaml의 StartupUri 속성에 MainWindow.xaml이 명시되어 있기 때문에 프로그램 실행시 메인 UI가 보이게 된다.

```
<Application x:Class="FirstWPF.App"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="clr-namespace: FirstWPF">
```

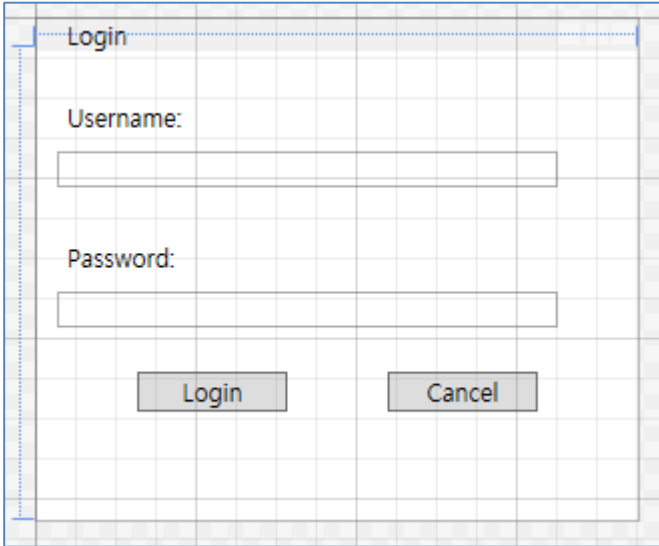
Advanced WPF Programming

WPF 프로젝트 구조 이해

- 메인 UI를 보여주기 전에 로그인 과정을 거치기위해 App.xaml.cs에 다음을 작성

```
public partial class App : Application
{
    protected override void OnStartup(StartupEventArgs e)
    {
        base.OnStartup(e);
        LoginWindow loginWindow = new LoginWindow();
        loginWindow.Show();
    }
}
```

WPF 프로젝트 구조 이해



```
<Window x:Class="FirstWPF.LoginWindow"
        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:FirstWPF"
        mc:Ignorable="d"
        Title="Login" Height="250" Width="300">
    <Grid>
        <Label Content="Username:" HorizontalAlignment="Left"
              VerticalAlignment="Top" Margin="10,20,0,0"/>
        <TextBox x:Name="UsernameTextBox" HorizontalAlignment="Left"
              VerticalAlignment="Top" Margin="10,50,0,0" Width="250"/>

        <Label Content="Password:" HorizontalAlignment="Left"
              VerticalAlignment="Top" Margin="10,90,0,0"/>
        <PasswordBox x:Name="PasswordBox" HorizontalAlignment="Left"
              VerticalAlignment="Top" Margin="10,120,0,0" Width="250"/>

        <Button Content="Login" HorizontalAlignment="Left" VerticalAlignment="Top"
              Margin="50,160,0,0" Width="75" Click="LoginButton_Click"/>
        <Button Content="Cancel" HorizontalAlignment="Right" VerticalAlignment="Top"
              Margin="0,160,50,0" Width="75" Click="CancelButton_Click"/>
    </Grid>
</Window>
```

```
public partial class LoginWindow : Window
{
    public bool IsAuthenticated { get; private set; }

    public LoginWindow()
    {
        InitializeComponent();
    }

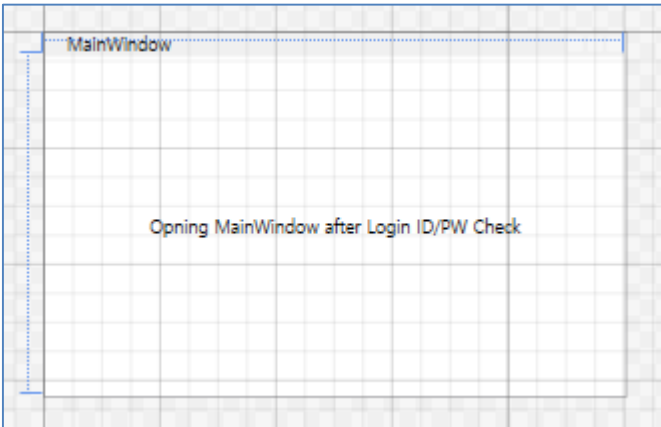
    private void LoginButton_Click(object sender, RoutedEventArgs e)
    {
        string username = UsernameTextBox.Text;
        string password = PasswordBox.Password;

        if (username == "admin" && password == "1234")
        {
            MainWindow mainWindow = new MainWindow();
            mainWindow.Show();
            this.Close();
        }
        else
        {
            MessageBox.Show("Invalid username or password.", "Login Failed", MessageBoxButton.OK, MessageBoxImage.Error);
        }
    }

    private void CancelButton_Click(object sender, RoutedEventArgs e)
    {
        this.Close();
    }
}
```

WPF 프로젝트 구조 이해

- 정상 인증이 끝난 후 실제 작업이 이뤄질 MainWindow

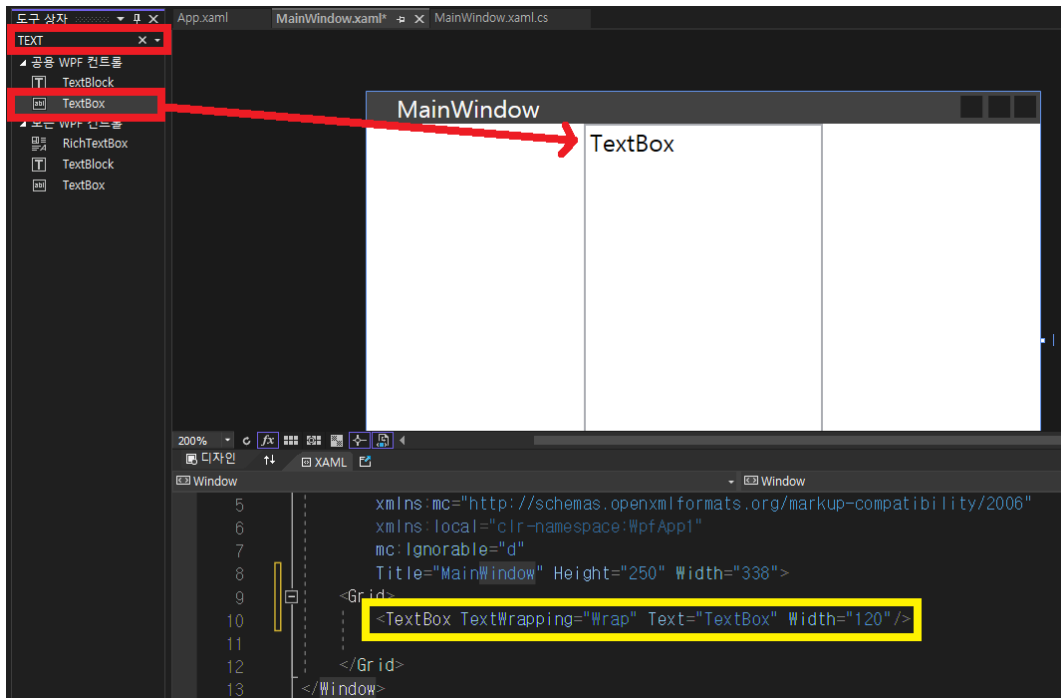


```
<Window x:Class="FirstWPF.MainWindow"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:local="clr-namespace:FirstWPF"
  mc:Ignorable="d"
  Title="MainWindow" Height="250" Width="400">
  <Grid>
    <Label Content="Opning MainWindow after Login ID/PW Check"
      HorizontalAlignment="Center" VerticalAlignment="Center"></Label>
  </Grid>
</Window>
```

Advanced WPF Programming

WPF Control

- Control 추가
 - Toolbox를 열어 원하는 컨트롤 추가
 - XAML 코드에서 직접 코드
 - CS파일에서 추가
 - 도구상자에서 검색 후 드래그 혹은 더블클릭디자인어에 컨트롤이 추가 되면 아래 XAML에도 자동으로 코드가 생성된다.



Advanced WPF Programming

WPF Control

- 윈도우 속성

- 어플리케이션UI를 이루는 기본적인 윈도우, 창에 대한 속성

- ◆ SizeToContent

- 윈도우도 Height나 Width를 통해 크기 입력이 가능하지만, 윈도우에 크기 설정을 하게 되면 경계선, 타이틀 바 등이 모두 포함된 크기라 실제 작업을 하는 내부 영역이 작아질 수 있기 때문에 내부의 최상위 Content에 크기 입력을 하고 윈도우는 SizeToContent를 WidthAndHeight 로 두게 되면 윈도우 크기가 Content에 딱 맞게 맞춰진다.

- ◆ ShowInTaskbar

- 윈도우가 뜨면 하단 작업 표시줄에 해당 윈도우가 표시되는 것을 보여줄지 여부를 설정
 - false 로 하면 작업표시줄에 표시가 되지 않는다.

- ◆ WindowStyle

- None 선택 시 테두리/타이틀 바가 없는 순수한 Content 영역만 표시된다.

- ◆ AllowTransparency

- 윈도우 자체에 투명도를 허용할 지 설정
 - True로 설정 시 윈도우도 배경색 지정 할 수 있는데, 이 때 배경을 투명(Transparent)나 투명도를 줬을 때, 투명한 윈도우가 만들어 진다. 이 속성을 위해서는 반드시 WindowStyle 속성이 None으로 되어있어야 한다.

- ◆ Resize

- None 선택 시 테두리/타이틀 바가 없는 순수한 Content 영역만 표시된다.

Advanced WPF Programming

WPF Control

- 기능별 WPF 컨트롤

레이아웃	Border, BulletDecorator, Canvas, DockPanel, Expander, Grid, GridView, GridSplitter, GroupBox, Panel, ResizeGrip, Separator, ScrollBar, ScrollViewer, StackPanel, Thumb, Viewbox, VirtualizingStackPanel, Window, WrapPanel
미디어	Image, MediaElement, SoundPlayerAction
메뉴	ContextMenu, Menu, ToolBar
탐색	Frame, Hyperlink, Page, NavigationWindow, TabControl
선택	CheckBox, ComboBox, ListBox, RadioButton, Slider

Advanced WPF Programming

WPF Control

- WPF 컨트롤
 - Winform에서 제공하는 기본 컨트롤
 - Button, textbox, Combobox, CheckBox, RadioButton
 - WPF 제공 컨트롤
 - TextBlock, InkCanvas, WindowsFormsHost 등
 - 순수 UI용 컨트롤 종류
 - Border, Rectangle, Ellipse, Line 등

Advanced WPF Programming

WPF Control

- WPF 컨트롤 공통 속성

속성	설명
Width	고정된 너비를 지정한다.
Height	고정된 높이를 지정한다.
MinWidth	가능한 최소 너비
MaxWidth	가능한 최대 너비
MinHeight	가능한 최소 높이
MaxHeight	가능한 최대 높이
HorizontalAlignment	요소가 사용 가능한 공간보다 작은 경우 수평 위치
VerticalAlignment	요소가 사용 가능한 공간보다 작은 수직 위치
Margin	요소 바깥쪽의 여백
Padding	요소 테두리와 요소 콘텐츠 사이의 여백
Visibility	필요한 경우 요소가 레이아웃 시스템 내에서 보이지 않도록 함
FlowDirection	텍스트의 방향
Panel.ZIndex	요소가 겹치는 경우의 우선순위
RenderTransform	레이아웃을 수정하지 않고 변경을 적용하는 속성
LayoutTransform	레이아웃에 영향을 주는 변경을 적용하는 속성

Advanced WPF Programming

WPF Control

- WPF 컨트롤 속성
 - 색상 관련 속성
 - 변경 가능한 색상의 종류 : 배경(Background), 글자(Foreground), 경계색(BorderBrush)
 - 경우에 따라 Fill, Stroke 사용
 - 컬러명을 직접 넣거나 (Red, Blue 등) ARGB 값 입력 (##FF3E232)
 - 폰트 관련 속성
 - 폰트 종류 (FontFamily), 크기(FontSize), 두께 설정 가능(FontWeight)
 - Content / Text 속성
 - Button이나 TextBox 같은 컨트롤에서는 문자를 컨트롤 안에 표시 가능
 - Button, CheckBox, RadioButton, Label TextBox 등 : 안에 들어가는 문자를 Content 속성으로 넣을 수 있다.
 - TextBox, TextBlock : Text 속성으로 넣어줘야 함.
 - Content 속성의 경우 다양한 컨트롤을 넣을 수 있으나 Text 속성은 Text(TextElement 컨트롤)만 넣을 수 있다.

Advanced WPF Programming

WPF Control

- IsEnabled / IsChecked / IsReadOnly / Focusable 속성
 - IsEnabled : 모든 컨트롤에 적용, 컨트롤의 활성화 유무 설정 (True / False)
 - IsChecked : CheckBox나 RadioButton, Toggle Button 같은 컨트롤에서 체크 선택 여부 설정
 - IsReadOnly : TextBox에서 사용. TextBox 내 텍스트를 읽기 전용으로 설정
 - Focusable : 모든 컨트롤에 적용. 컨트롤 자체에 포커스를 줄 수 있는지(선택 할 수 있는지) 여부를 설정하는 속성
- Opacity / OpacityMask
 - 투명도와 관련된 속성
 - Opacity : 컨트롤 자체의 투명도 설정
 - OpacityMask : 컨트롤 위에 덮어버리는 창 수정이 불가능한 이미지 파일을 가지고 작업 시 유용하게 사용되지만 Opacity 속성만으로 유사한 효과를 낼 수 있다.

Advanced WPF Programming

WPF Control

- 컨트롤 변경
 - 개별 컨트롤들은 Transform을 통해 형태 변경 가능
 - ex) Button 하위에 RenderTransform 이라는 속성을 정의하면 여러개의 Transform을 볼 수 있다.
 - ScaleTransform : 컨트롤의 크기 변경. (ScaleX, ScaleY를 통해 변환)
 - CenterX와 CenterY는 변환의 기준점
 - RotateTransform : 컨트롤 회전 (Angle을 통해 각도 조절)
 - CneterX와 CenterY를 통해 중심축 설정
 - SkewTransform : X,Y 방향으로 컨트롤의 형태를 기울임
 - AngleX와 AngleY를 통해 설정
 - RanderTransform VS LayoutTransform
 - RanderTransform : 컨트롤이 화면상에 그려지는 형태만 변형
 - LayoutTransform " 컨트롤이 차지하고있는 Layout 즉 공간 자체를 변형

Advanced WPF Programming

WPF Control

- 기능별 WPF 컨트롤

버튼	Button, RepeatButton
데이터 표시	DataGrid, ListView, TreeView.
날짜 표시 및 선택	Calendar, DatePicker
대화 상자	OpenFileDialog, PrintDialog, SaveFileDialog
디지털 잉크	InkCanvas, InkPresenter
문서	DocumentViewer, FlowDocumentPageViewer, FlowDocumentReader, FlowDocumentScrollViewer, StickyNoteControl
입력	TextBox, RichTextBox, PasswordBox
사용자 정보	AccessText, Label, Popup, ProgressBar, StatusBar, TextBlock, ToolTip

Advanced WPF Programming

WPF Control

Advanced WPF Programming

WPF Control

Advanced WPF Programming

WPF Control

- 주요 Control에서 사용 빈도가 높은 이벤트 종류

Control	주요 이벤트 종류	설명
Button	Click	버튼 클릭 시 발생
	MouseEnter	마우스가 버튼 위로 올라갔을 때 발생
	MouseLeave	마우스가 버튼에서 벗어났을 때 발생
TextBox	TextChanged	텍스트가 변경될 때 발생
	GotFocus	텍스트 박스가 포커스를 얻었을 때 발생
	LostFocus	텍스트 박스가 포커스를 잃었을 때 발생
CheckBox	Checked	체크박스가 체크되었을 때 발생
	Unchecked	체크박스가 체크 해제되었을 때 발생
	Click	체크박스가 클릭되었을 때 발생

Advanced WPF Programming

WPF Control

- 주요 Control에서 사용 빈도가 높은 이벤트 종류

Control	주요 이벤트 종류	설명
RadioButton	Checked	라디오 버튼이 선택되었을 때 발생
	Unchecked	라디오 버튼이 선택 해제되었을 때 발생
	Click	라디오 버튼이 클릭되었을 때 발생
ListView	SelectionChanged	선택 항목이 변경될 때 발생
	MouseDoubleClick	항목이 더블 클릭되었을 때 발생
	ItemContainerGenerator.StatusChanged	항목 컨테이너 상태 변경 시 발생
ListBox	SelectionChanged	선택 항목이 변경될 때 발생
	MouseDoubleClick	항목이 더블 클릭되었을 때 발생
	PreviewMouseDown	마우스 버튼이 눌렸을 때 발생 (미리보기)
DataGrid	SelectionChanged	선택 항목이 변경될 때 발생
	CellEditEnding	셀 편집이 종료될 때 발생
	RowEditEnding	행 편집이 종료될 때 발생

Advanced WPF Programming

WPF Control

Advanced WPF Programming

WPF Window, Page, App.xaml 연동

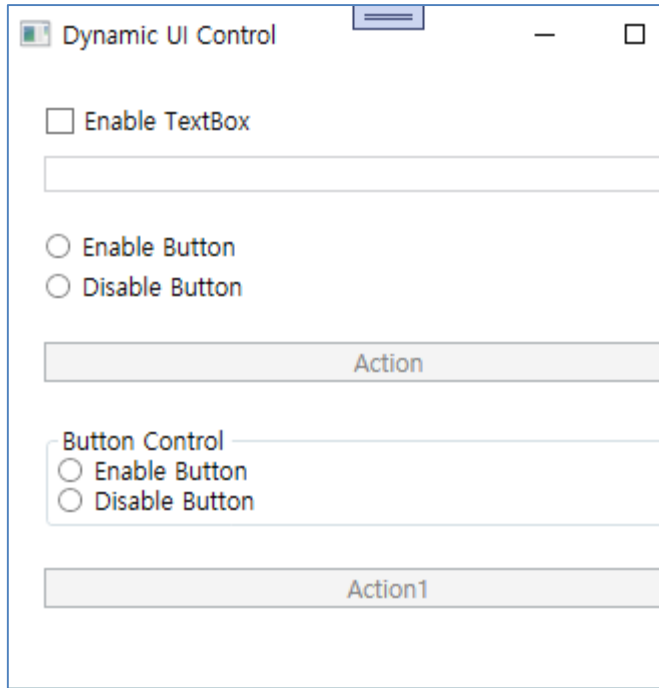
- 이벤트 처리
 - window 에서 Page로 이동할 때의 방법

```
private void Button_Click_1(object sender, RoutedEventArgs e)
{
    Page1 page = new Page1();
    page.Title = "testPage";
    this.Content = page;
}
```

Advanced WPF Programming

WPF Control 활용 및 Event 예

- 이벤트를 통해 컨트롤의 속성 제어 방법
 - CheckBox 선택 여부에 따라 TextBox 사용 가능 결정
 - Radio 버튼 또는 GroupBox 내의 Radio 버튼 선택에 따라 Button의 활성화/비활성화 설정



```
<Grid>
  <StackPanel Margin="20">
    <CheckBox x:Name="EnableTextBoxCheckBox" Content="Enable TextBox"
      Checked="EnableTextBoxCheckBox_Checked" Unchecked="EnableTextBoxCheckBox_Unchecked" />
    <TextBox x:Name="InputTextBox" IsEnabled="False" Margin="0,10,0,0" />

    <RadioButton x:Name="EnableButtonRadioButton" Content="Enable Button"
      Checked="EnableButtonRadioButton_Checked" Margin="0,20,0,0" />
    <RadioButton x:Name="DisableButtonRadioButton" Content="Disable Button"
      Checked="DisableButtonRadioButton_Checked" Margin="0,5,0,0" />

    <Button x:Name="ActionButton" Content="Action" IsEnabled="False" Margin="0,20,0,0" />

    <GroupBox Header="Button Control" Margin="0,20,0,0">
      <StackPanel>
        <RadioButton Content="Enable Button" Checked="RadioButton_Checked" />
        <RadioButton Content="Disable Button" Checked="RadioButton_Checked" />
      </StackPanel>
    </GroupBox>

    <Button x:Name="ActionButton1" Content="Action1" IsEnabled="False" Margin="0,20,0,0" />
  </StackPanel>
</Grid>
```

Advanced WPF Programming

WPF Control 활용 및 Event 예

```
private void EnableTextBoxCheckBox_Checked(object sender, RoutedEventArgs e)
{
    InputTextBox.IsEnabled = true; // 체크박스가 체크되면 텍스트박스를 활성화
}

private void EnableTextBoxCheckBox_Unchecked(object sender, RoutedEventArgs e)
{
    InputTextBox.IsEnabled = false; // 체크박스가 체크 해제되면 텍스트박스를 비활성화
}
```

```
private void EnableButtonRadioButton_Checked(object sender, RoutedEventArgs e)
{
    ActionButton.IsEnabled = true; // 라디오 버튼이 체크되면 버튼을 활성화
}

private void DisableButtonRadioButton_Checked(object sender, RoutedEventArgs e)
{
    ActionButton.IsEnabled = false; // 다른 라디오 버튼이 체크되면 버튼을 비활성화
}
```

Advanced WPF Programming

WPF Control 활용 및 Event 예

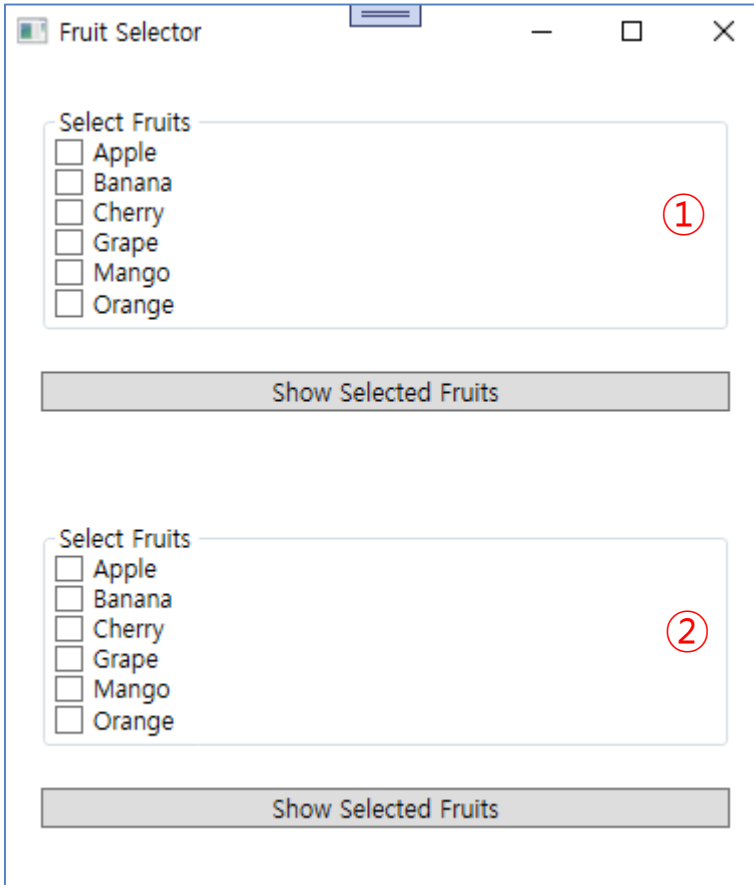
```
private void RadioButton_Checked(object sender, RoutedEventArgs e)
{
    // 어떤 라디오 버튼이 체크되었는지 확인
    if (sender is RadioButton radioButton)
    {
        ActionButton1.IsEnabled = radioButton.Content.ToString() == "Enable Button";
        // 버튼 활성화 여부 결정
    }
}
```

- GroupBox는 여러 개의 Controls을 하나의 그룹으로 표현하기 좋은 컨트롤이다.

Advanced WPF Programming

WPF Control 활용 및 Event 예

- GroupBox에 여러 개의 CheckBox가 있고 여러 개의 항목을 선택할 수 있도록 한 후 선택한 모든 종류를 출력하도록 하는 경우



```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition></RowDefinition>
    <RowDefinition></RowDefinition>
  </Grid.RowDefinitions>
  <StackPanel Grid.Row="0" Margin="20">
    <GroupBox Header="Select Fruits" Margin="0,0,0,20">
      <StackPanel>
        <CheckBox x:Name="AppleCheckBox" Content="Apple" />
        <CheckBox x:Name="BananaCheckBox" Content="Banana" />
        <CheckBox x:Name="CherryCheckBox" Content="Cherry" />
        <CheckBox x:Name="GrapeCheckBox" Content="Grape" />
        <CheckBox x:Name="MangoCheckBox" Content="Mango" />
        <CheckBox x:Name="OrangeCheckBox" Content="Orange" />
      </StackPanel>
    </GroupBox>

    <Button Content="Show Selected Fruits" Click="ShowSelectedFruitsButton_Click"/>
  </StackPanel>
</Grid>
```


Advanced WPF Programming

WPF Control 활용 및 Event 예

```
<StackPanel Grid.Row="1" Margin="20">
    <GroupBox Header="Select Fruits" Margin="0,0,0,20">
        <ItemsControl x:Name="FruitCheckBoxes">
            <ItemsControl.ItemsPanel>
                <ItemsPanelTemplate>
                    <StackPanel />
                </ItemsPanelTemplate>
            </ItemsControl.ItemsPanel>
        </ItemsControl>
    </GroupBox>

    <Button Content="Show Selected Fruits" Click="ShowSelectedFruitsButton_Click1"/>
</StackPanel>
```

②

Advanced WPF Programming

WPF Control 활용 및 Event 예

```
private void ShowSelectedFruitsButton_Click(object sender, RoutedEventArgs e) ①
{
    StringBuilder selectedFruits = new StringBuilder("Selected Fruits:\n");

    if (AppleCheckBox.IsChecked == true) selectedFruits.AppendLine(AppleCheckBox.Content.ToString());
    if (BananaCheckBox.IsChecked == true) selectedFruits.AppendLine(BananaCheckBox.Content.ToString());
    if (CherryCheckBox.IsChecked == true) selectedFruits.AppendLine(CherryCheckBox.Content.ToString());
    if (GrapeCheckBox.IsChecked == true) selectedFruits.AppendLine(GrapeCheckBox.Content.ToString());
    if (MangoCheckBox.IsChecked == true) selectedFruits.AppendLine(MangoCheckBox.Content.ToString());
    if (OrangeCheckBox.IsChecked == true) selectedFruits.AppendLine(OrangeCheckBox.Content.ToString());

    MessageBox.Show(selectedFruits.ToString(), "Fruit Selection");
}
```

Advanced WPF Programming

WPF Control 활용 및 Event 예

```
private List<string> fruits = new List<string> { "Apple", "Banana", "Cherry", "Grape", "Mango", "Orange" };

public GroupBoxMultiCheckBoxWindow()
{
    InitializeComponent();
    CreateCheckBoxes();
}

private void CreateCheckBoxes()
{
    foreach (var fruit in fruits)
    {
        CheckBox checkBox = new CheckBox { Content = fruit };
        FruitCheckBoxes.Items.Add(checkBox);
    }
}
```

②

Advanced WPF Programming

WPF Control 활용 및 Event 예

```
private void ShowSelectedFruitsButton_Click1(object sender, RoutedEventArgs e)
{
    StringBuilder selectedFruits = new StringBuilder("Selected Fruits:\n");

    foreach (CheckBox checkBox in FruitCheckBoxes.Items)
    {
        if (checkBox.IsChecked == true)
        {
            selectedFruits.AppendLine(checkBox.Content.ToString());
        }
    }

    MessageBox.Show(selectedFruits.ToString(), "Fruit Selection");
}
```

②

Advanced WPF Programming

WPF Window, Page, App.xaml 연동

- window 에서 window로 이동할 때의 방법
 - window 에서 더블 클릭 후 Show()나 ShowDialog() 메서드를 이용하여 창을 이동
 - Show : 모달리스 창. 뒤의 부모 폼으로 이동 가능
 - ShowDialog : 모달 창. 뒤의 부모 폼으로 이동이 불가능하다. 창을 닫을 때 까지 부모창 또는 다른 창으로 이동이 불가

```
private void hand_click(object sender, RoutedEventArgs e)
{
    Window w = new Window();
    w.Title = "TestDialog";
    w.Show();
}
```

```
private void HelloClick(object sender, RoutedEventArgs e)
{
    Window1 w1 = new Window1();
    w1.Title = "Hello Test";
    w1.ShowDialog();
}
```

Advanced WPF Programming

개인 UI 구상

Advanced WPF Programming

개인 UI 구상

Advanced WPF Programming

개인 UI 구상

[C# 기반]

Advanced WPF Programming

WPF RESOURCE

Advanced WPF Programming

WPF Resource

- Resource

- WPF에서는 원하는 모든 데이터를 리소스 형태로 저장하여 컨트롤이나 윈도우를 지역적으로 저장하거나 전체 애플리케이션을 전역으로 저장할 수 있다.
 - 데이터란 실제 정보일 수도 있고 WPF 컨트롤들의 계층 구조일 수도 있다.
- 리소스는 앱의 여러 위치에서 다시 사용할 수 있는 개체이며 브러시와 스타일이 있습니다.
- 리소스는 **x:Key** 속성으로 키를 받는다.
 - 키를 이용해 애플리케이션 내 다른 위치에 있는 값을 참조할 수 있다.
- 키는 markup extension인 **StaticResource**와 결합되어 있다.

Advanced WPF Programming

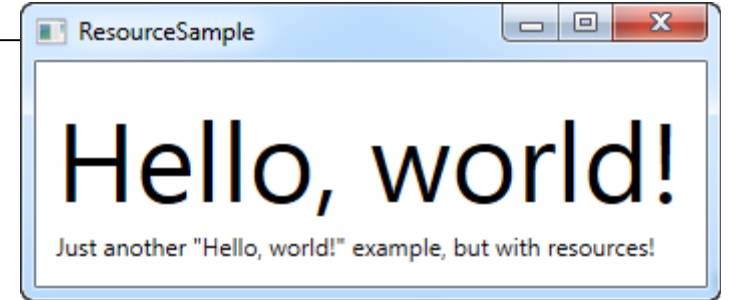
WPF Resource

- Window에 리소스 설정 및 사용 방법

```
<Window x:Class="WpfTutorialSamples.WPF_Application.ResourceSample"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:sys="clr-namespace:System;assembly=mscorlib"
  Title="ResourceSample" Height="150" Width="350">

  <Window.Resources>
    <sys:String x:Key="strHelloWorld">Hello, world!</sys:String>
  </Window.Resources>

  <StackPanel Margin="10">
    <TextBlock Text="{StaticResource strHelloWorld}" FontSize="56" />
    <TextBlock>Just another "<TextBlock Text="{StaticResource strHelloWorld}" />"
      example, but with resources!
    </TextBlock>
  </StackPanel>
</Window>
```



Advanced WPF Programming

WPF Resource

- StaticResource vs. DynamicResource

- **StaticResource**의 경우 XAML이 로드될 때 단 한번 할당된다는 것이 주요 특징이다.
- **StaticResource**를 사용하면 나중에 리소스가 변경되더라도 변경사항이 반영되지 않는다.
- **DynamicResource**는 실제 필요할 때마다 할당한다. 리소스가 변경될 때 마다 반복해서 할당할 수 있다.
- **DynamicResource**는 디자인할 때 존재하지 않았던 리소스들도 사용할 수 있다.

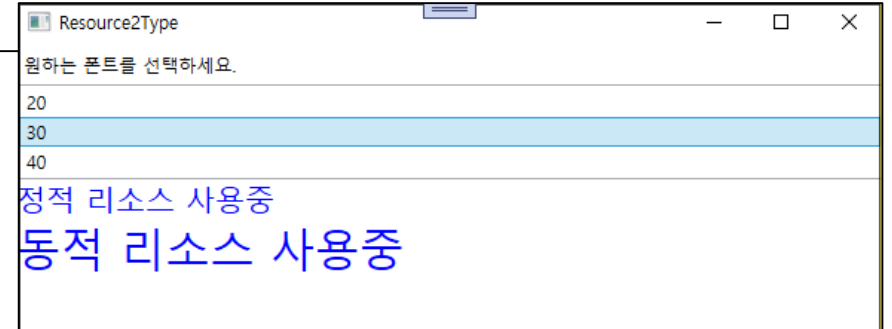
정적 리소스	동적 리소스
처음으로 참조한 리소스의 값을 변경할 의도가 없는 경우	런타임 시에 값을 알 수 있는 경우
페이지나 응용 프로그램에 리소스를 모아 주로 사용할 경우	사용자 지정 컨트롤에 대한 테마 스타일을 만들거나 참조하는 경우
	응용 프로그램 수명 동안 ResourceDictionary의 내용을 조정하려는 경우
	참조된 리소스가 즉시 사용되지 않을 경우
	setter 값을 테마 또는 다른 사용자 설정의 영향을 받는 다른 값에서 가져올 수 있는 스타일을 만드는 경우

Advanced WPF Programming

WPF Resource

- 정적 리소스와 동적 리소스 적용 차이

```
<Window.Resources>
  <Style TargetType="TextBlock" x:Key="FontStyle">
    <Setter Property="FontSize" Value="20" />
    <Setter Property="Foreground" Value="Blue" />
  </Style>
</Window.Resources>
<StackPanel>
  <Label>원하는 폰트를 선택하세요.</Label>
  <ListBox Name="lbox_fsize" SelectionChanged="listBox_SelectionChanged">
    <ListBoxItem Content="20" />
    <ListBoxItem Content="30" />
    <ListBoxItem Content="40" />
  </ListBox>
  <TextBlock Name="sbox" Style="{StaticResource FontStyle}" Text="정적 리소스 사용중" />
  <TextBlock Name="dbox" Style="{DynamicResource FontStyle}" Text="동적 리소스 사용중" />
</StackPanel>
```



Advanced WPF Programming

WPF Resource

- 정적 리소스와 동적 리소스 적용 차이

```
private void listBox_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    int cnt = 0;
    Style mystyle = Resources["FontStyle"] as Style;
    Style chstyle = new Style();
    ListBoxItem lvi = lbox_fsize.SelectedItem as ListBoxItem;

    double fsize = double.Parse(lvi.Content.ToString());

    foreach (Setter s in mystyle.Setters)
    {
        if (cnt == 0)
        {
            chstyle.Setters.Add(new Setter(s.Property, fsize));
        }
        else
        {
            chstyle.Setters.Add(new Setter(s.Property, s.Value));
        }
        cnt++;
    }
    Resources["FontStyle"] = chstyle;
}
```

Advanced WPF Programming

WPF Resource

- Local Resource와 Application Resource
 - 하나의 특정 컨트롤을 위한 리소스가 필요할 경우 윈도우가 아닌 컨트롤에 추가해서 지역적으로 사용할 수 있다.
 - 기존 윈도우의 모든 부분에서 리소스에 접근할 수 있는 것에 대해 작동 방식은 동일하나 유일한 차이점은 **생성된 컨트롤 범위 내에서만 접근할 수 있다는 것이다.**
 - WPF는 주어진 리소스를 찾기 위해 자동으로 로컬 컨트롤에서 윈도우로, App.xaml 로 범위를 올려간다.
 - 코드 비하인드에서도 리소스에 접근할 수 있다.
 - **FindResource()** 메서드를 사용해 리소스를 찾으면 object로 리턴 된다.

Advanced WPF Programming

WPF Resource Resource 적용 범위

```
<StackPanel Margin="10">
  <StackPanel.Resources>
    <sys:String x:Key="ComboBoxTitle">Items:</sys:String>
  </StackPanel.Resources>
  <Label Content="{StaticResource ComboBoxTitle}" />
</StackPanel>
```

```
<Window x:Class="WpfSamples.WPF_Application.ResourceSample"
...
  <Window.Resources>
    <sys:String x:Key="strHelloWorld">Hello, world!</sys:String>
  </Window.Resources>
  <StackPanel Margin="10">
    <TextBlock Text="{StaticResource strHelloWorld}" FontSize="56" />
    <TextBlock>Just another "<TextBlock Text="{StaticResource strHelloWorld}" />" example, but with resources!</TextBlock>
  </StackPanel>
</Window>
```

```
<Application x:Class="WpfSamples.App"
...
  <Application.Resources>
    <sys:String x:Key="ComboBoxTitle">Items:</sys:String>
  </Application.Resources>
</Application>
```


Advanced WPF Programming

WPF Resource

```
<Application x:Class="WpfTutorialSamples.App"
...
<Application.Resources>
  <sys:String x:Key="strApp">Hello, Application world!</sys:String>
</Application.Resources>
</Application>
```

```
<Window x:Class="WpfTutorialSamples.WPF_Application.ResourcesFromCodeBehindSample"
...
<Window.Resources>
  <sys:String x:Key="strWindow">Hello, Window world!</sys:String>
</Window.Resources>
<DockPanel Margin="10" Name="pnlMain">
  <DockPanel.Resources>
    <sys:String x:Key="strPanel">Hello, Panel world!</sys:String>
  </DockPanel.Resources>

  <WrapPanel DockPanel.Dock="Top" HorizontalAlignment="Center" Margin="10">
    <Button Name="btnClickMe" Click="btnClickMe_Click">Click me!</Button>
  </WrapPanel>

  <ListBox Name="lbResult" />
</DockPanel>
</Window>
```

Hello, Panel world!
Hello, Window world!
Hello, Application world!

```
using System;
using System.Windows;

namespace WpfTutorialSamples.WPF_Application
{
    public partial class ResourcesFromCodeBehindSample : Window
    {
        public ResourcesFromCodeBehindSample()
        {
            InitializeComponent();
        }

        private void btnClickMe_Click(object sender, RoutedEventArgs e)
        {
            lbResult.Items.Add(pnlMain.FindResource("strPanel").ToString());
            lbResult.Items.Add(this.FindResource("strWindow").ToString());
            lbResult.Items.Add(Application.Current.FindResource("strApp").ToString());
        }
    }
}
```

Advanced WPF Programming

WPF 데이터 바인딩 개요

- 바인딩이란
 - 두 개 이상의 객체 간 속성을 연결하여 한 객체의 속성이 변경되었을 때 바인딩 된 다른 객체의 속성 또한 변경되도록 하는 작업.
 - 예) chekBox 에 체크가 되었을 때에는 TextBox가 활성화 되도록 구현하는 경우 체크 여부를 설정하는 IsChecked 라는 속성과 TextBox의 IsEnabled 라는 속성을 바인딩 처리
- 바인딩의 방향성
 1. OneWay Binding : 소스가 변경될 때마다 타깃이 갱신된다.
 2. TwoWay Binding : 타깃이나 소스 둘 중에 한 쪽이 변경되면 서로 갱신한다.
 3. OneWayToSource : 타깃이 변경될 때마다 소스가 갱신된다.
 4. OneTime : 바인딩 클래스가 인스턴스화될 때 한 번 타깃에 반영되고 그 뒤로 소스가 변경되어도 타깃에 반영되지 않는다. 물론 타깃이 변경되어도 소스에 반영되지 않는다.

[바인딩 참고]

<https://learn.microsoft.com/ko-kr/dotnet/desktop/wpf/data/?view=netdesktop-7.0>

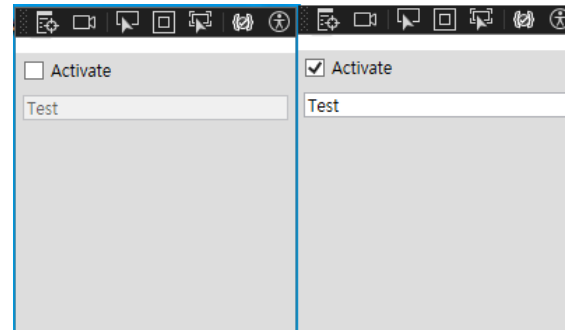
Advanced WPF Programming

WPF 데이터 바인딩 개요

- 바인딩 처리

```
<CheckBox Name="chbx" Content="Activate" Margin="5"
    IsChecked="{Binding ElementName=chbx, Path=IsChecked, Mode=OneWay}"/>
<TextBox Name="txtb" Text="Test" Margin="5"/>
```

- 데이터의 흐름이 Two-Way 일 때에는 어느 쪽에서 바인딩을 걸어도 상관이 없지만 One-Way인 경우 바인딩을 거는 위치에 따라 결과가 달라진다.
- One-Way가 되면 바인딩 작업을 한 객체의 속성은 반대편 객체의 속성에 따라 변하지만, 반대편 객체의 속성은 바인딩 작업을 한 객체의 속성은 무시한다.
- 즉, One-Way일 때 바인딩 된 속성이 뭐가 되든 신경 안쓰기 때문에 CheckBox에서는 TextBox가 활성화 될 때에는 체크가 되지만, 반대로 체크를 한다고 TextBox가 활성화 되지는 않는 것.

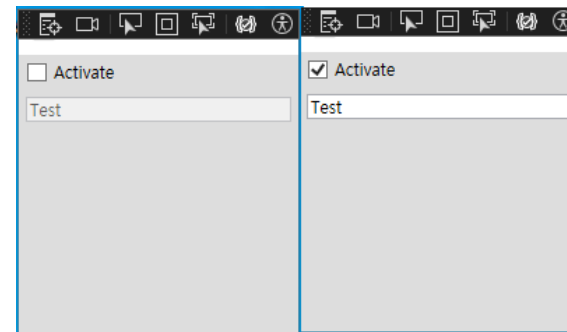


Advanced WPF Programming

WPF 데이터 바인딩 개요

- 바인딩 처리

```
<StackPanel Height="200" Width="200">
    <CheckBox Name="chbx" Content="Activate" Margin="5" />
    <!-- 바인딩처리 -->
    <TextBox Name="txtb" Text="Test" Margin="5"
        IsEnabled="{Binding ElementName=chbx, Path=IsChecked}" />
</StackPanel>
```



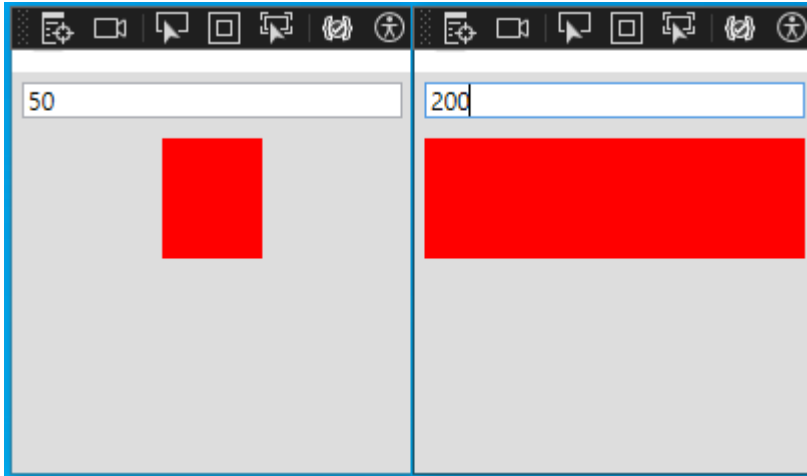
- TextBox의 IsEnabled 속성이 CheckBox의 IsChecked 라는 속성에 따라 변경되는 것
이를 위해 TextBox의 IsEnabled 라는 속성에 바인딩을 설정 해 준다.
 - Binding : 바인딩 동작을 알린다.
 - ElementName : 바인딩 해야 할 객체를 지정한다.
 - , : 속성이 변경됨을 알린다.
 - Path : 이후에 객체의 속성을 설정한다.
- 바인딩은 CheckBox의 isChecked 에 걸어도 TextBox의 Isenabled에 걸어도 상관이 없다.

Advanced WPF Programming

WPF 데이터 바인딩 개요

- Bool 타입의 데이터가 아니라 다른 데이터들도 바인딩이 가능하다.
또한, XAML 상에서 바인딩을 할 때 데이터 타입은 무관하다.
- XAML 에서 바인딩을 할 때 신경써야 할 부분은 데이터 자체의 타입이 아니라 데이터 자체의 형태가 된다.

```
<StackPanel Height="200" Width="200">  
  <TextBox Name="txtb" Text="50" Margin="5" TextChanged="txtb_TextChanged" />  
  <Border Margin="5" Height="60" Width="{Binding ElementName=txtb, Path=Text}" Background="Red" />  
</StackPanel>
```

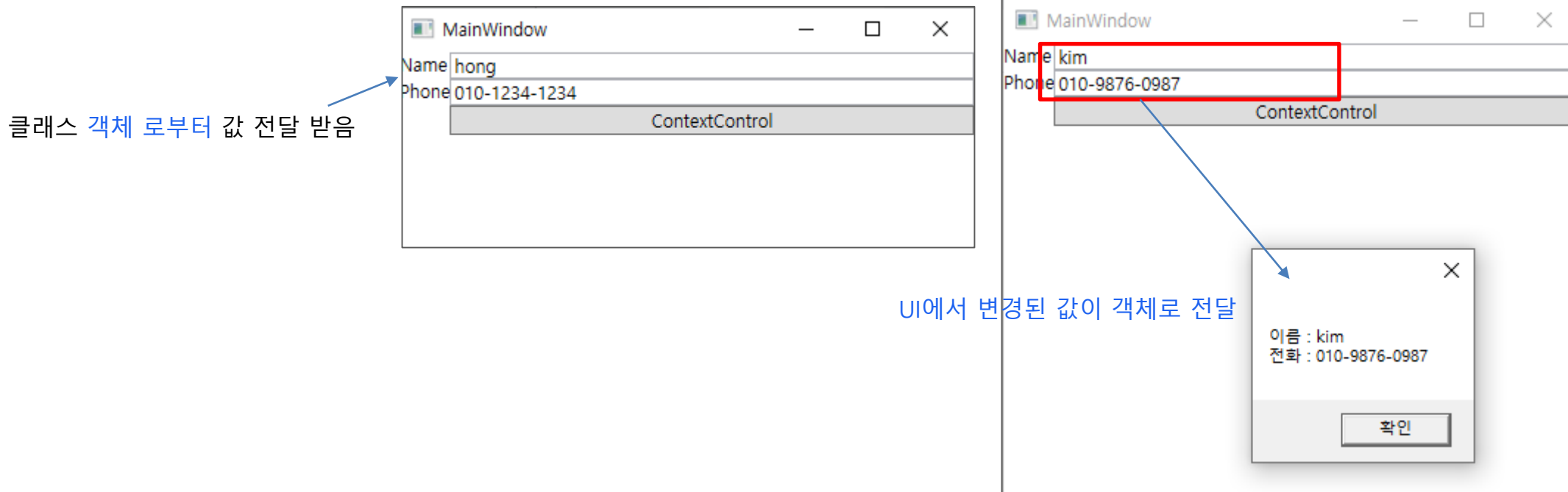


Text는 string타입이지만 double 타입인 Width와 바인딩

Advanced WPF Programming

WPF 데이터 바인딩 개요(DataContext 이용)

- DataContext를 이용한 데이터 바인딩
 - UI Control이 아닌 직접 작성한 클래스 객체 정보를 xaml 태그에 출력할 때 이용
 - 양방향 binding이 가능



Advanced WPF Programming

WPF 데이터 바인딩 개요 (DataContext 이용)

MainWindow.xaml

```
<Window x:Class="DataContextBind.MainWindow"
...
<Grid x:Name="Grid">
  <Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="Auto"/>
  </Grid.RowDefinitions>
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto"/>
    <ColumnDefinition Width="*/>
  </Grid.ColumnDefinitions>
  <TextBlock Grid.Column="0" Grid.Row="0">Name</TextBlock>
  <TextBlock Grid.Column="0" Grid.Row="1">Phone</TextBlock>
  <TextBox x:Name="TextBox1" Grid.Column="1" Grid.Row="0" Text="{Binding MemberName}"></TextBox>
  <TextBox x:Name="TextBox2" Grid.Column="1" Grid.Row="1" Text="{Binding MemberPhone}"></TextBox>
  <Button Grid.Column="1" Grid.Row="2" Name="button1" Click="OnClicked">ContextControl</Button>
</Grid>
</Window>
```

Advanced WPF Programming

WPF 데이터 바인딩 개요 (DataContext 이용)

MainWindow.xaml.cs

```
public partial class MainWindow : Window
{
    public MainWindow()
    {
        InitializeComponent();

        Member member = new Member()
        {
            MemberName = "hong",
            MemberPhone="010-1234-1234"
        };
        this.DataContext = member;
    }

    public void OnClicked(object sender, RoutedEventArgs e)
    {
        Member tmp = this.DataContext as Member;
        MessageBox.Show("이름 : " + tmp.MemberName + "\n전화 : " + tmp.MemberPhone);
    }
}
```

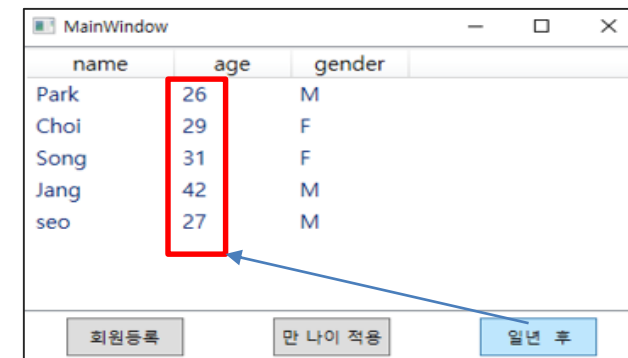
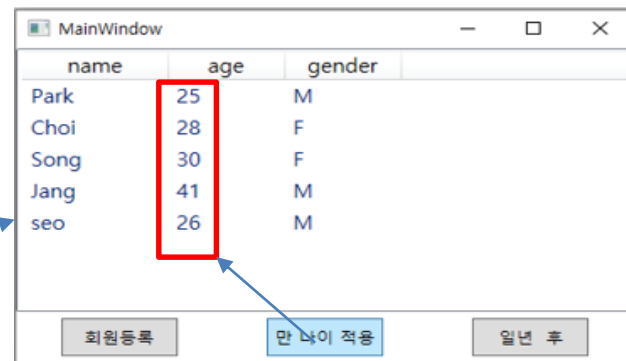
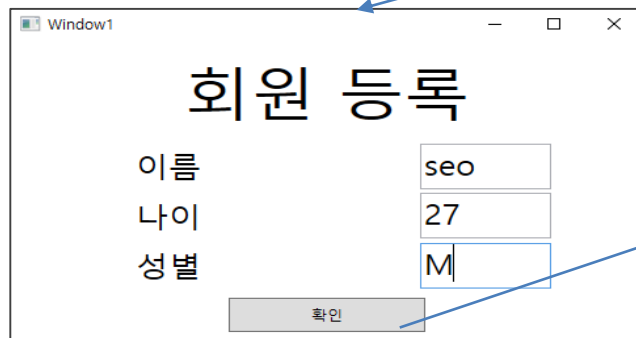
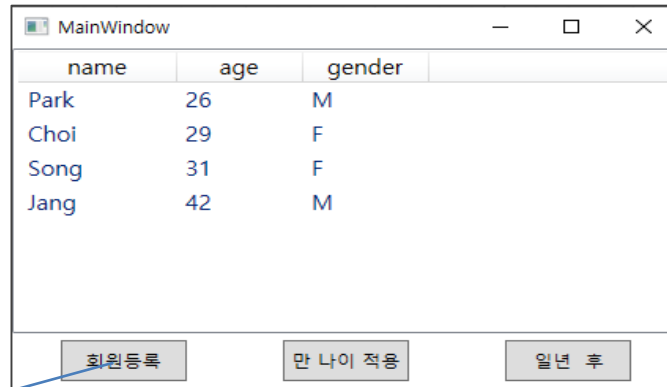
Member.cs

```
class Member
{
    public int Id { get; set; }
    public string MemberName { get; set; }
    public string MemberPhone { get; set; }
}
```


Advanced WPF Programming

WPF 데이터 바인딩 개요 (INotifyPropertyChanged.PropertyChanged 사용)

- ViewModel 변경 값을 UI에 적용



Advanced WPF Programming

WPF 데이터 바인딩 개요 (INotifyPropertyChanged.PropertyChanged 사용)

- MainWindow.xaml

```
<Window x:Class="InotifyPropertyBind.MainWindow"
...
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="5*"></RowDefinition>
    <RowDefinition Height="1*"></RowDefinition>
  </Grid.RowDefinitions>
  <ListView Grid.Row="0" Name="memberListView" FontSize="15">
    <ListView.View>
      <GridView>
        <GridViewColumn Header="name" DisplayMemberBinding="{Binding name}" Width="100"/>
        <GridViewColumn Header="age" DisplayMemberBinding="{Binding Age}" Width="80"/>
        <GridViewColumn Header="gender" DisplayMemberBinding="{Binding gender}" Width="80"/>
      </GridView>
    </ListView.View>
  </ListView>
  <Grid Grid.Row="1">
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="1*"></ColumnDefinition>
      <ColumnDefinition Width="1*"></ColumnDefinition>
      <ColumnDefinition Width="1*"></ColumnDefinition>
    </Grid.ColumnDefinitions>
    <Button Grid.Column="0" Name="AddMemberBtn" Width="80" Height="30" Content="회원등록" Click="AddMemberBtn_Click"></Button>
    <Button Grid.Column="1" Name="MinusOneYearBtn" Width="80" Height="30" Content="만 나이 적용" Click="MinusOneYearBtn_Click"></Button>
    <Button Grid.Column="2" Name="OneYearAfterBtn" Width="80" Height="30" Content="일년 후" Click="OneYearAfterBtn_Click"></Button>
  </Grid>
</Grid>
</Window>
```

Advanced WPF Programming

WPF 데이터 바인딩 개요(INotifyPropertyChanged/INotifyCollectionChanged 사용)

MainWindow.xaml.cs

```
public partial class MainWindow : Window
{
    ObservableCollection<Member> members = null;

    public MainWindow()
    {
        InitializeComponent();
        members = new ObservableCollection<Member>();
        members.Add(new Member() { name = "Park", Age = 26, gender = "M" });
        members.Add(new Member() { name = "Choi", Age = 29, gender = "F" });
        members.Add(new Member() { name = "Song", Age = 31, gender = "F" });
        members.Add(new Member() { name = "Jang", Age = 42, gender = "M" });
        memberListView.ItemsSource = members;
    }

    private void AddMemberBtn_Click(object sender, RoutedEventArgs e)
    {
        AddMemberWindow addMemberWindow = new AddMemberWindow();
        if (addMemberWindow.ShowDialog() == true)
        {
            string newName = addMemberWindow.nameBox.Text;
            int newAge = int.Parse(addMemberWindow.ageBox.Text);
            string newGender = addMemberWindow.genderBox.Text;

            members.Add(new Member() { name = newName, Age = newAge, gender = newGender });
        }
    }
}
```

Advanced WPF Programming

WPF 데이터 바인딩 개요(INotifyPropertyChanged/INotifyCollectionChanged 사용)

MainWindow.xaml.cs 계속

```
private void OneYearAfterBtn_Click(object sender, RoutedEventArgs e)
{
    foreach (var item in members)
    {
        item.Age++;
    }
}

private void MinusOneYearBtn_Click(object sender, RoutedEventArgs e)
{
    foreach (var item in members)
    {
        item.Age--;
    }
}
```

Advanced WPF Programming

WPF 데이터 바인딩 개요(INotifyPropertyChanged/INotifyCollectionChanged 사용)

- AddMemberWindow.xaml

```
<Window x:Class="InotifyPropertyBind.AddMemberWindow"
...
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="2*"></RowDefinition>
    <RowDefinition Height="1*"></RowDefinition>
    <RowDefinition Height="1*"></RowDefinition>
    <RowDefinition Height="1*"></RowDefinition>
    <RowDefinition Height="1*"></RowDefinition>
  </Grid.RowDefinitions>

  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="1*"></ColumnDefinition>
    <ColumnDefinition Width="1*"></ColumnDefinition>
  </Grid.ColumnDefinitions>
  <Grid Grid.Row="0" Grid.ColumnSpan="2">
    <TextBlock Text="회원 등록" VerticalAlignment="Center" HorizontalAlignment="Center" FontSize="50"></TextBlock>
  </Grid>
  <TextBlock Text="이름" Grid.Row="1" Grid.Column="0" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="25"></TextBlock>
  <TextBlock Text="나이" Grid.Row="2" Grid.Column="0" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="25"></TextBlock>
  <TextBlock Text="성별" Grid.Row="3" Grid.Column="0" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="25"></TextBlock>
  <TextBox Grid.Row="1" Grid.Column="1" Width="100" Height="40" Name="nameBox" FontSize="25"></TextBox>
  <TextBox Grid.Row="2" Grid.Column="1" Width="100" Height="40" Name="ageBox" FontSize="25"></TextBox>
  <TextBox Grid.Row="3" Grid.Column="1" Width="100" Height="40" Name="genderBox" FontSize="25"></TextBox>

  <Button Grid.Row="4" Grid.ColumnSpan="2" Width="150" Height="30" Content="확인" Click="Button_Click"></Button>
</Grid>
</Window>
```

Advanced WPF Programming

WPF 데이터 바인딩 개요(INotifyPropertyChanged/INotifyCollectionChanged 사용)

- AddMemberWindow.xaml.cs

```
public partial class AddMemberWindow : Window
{
    public AddMemberWindow()
    {
        InitializeComponent();
    }

    private void Button_Click(object sender, RoutedEventArgs e)
    {
        this.DialogResult = true;
    }
}
```

Advanced WPF Programming

관리할 데이터 설계

Advanced WPF Programming

관리할 데이터 설계

Advanced WPF Programming

관리할 데이터 설계

[C# 기반]

Advanced WPF Programming

WPF TRIGGER

Advanced WPF Programming

WPF Trigger 이해

- Trigger란?
 - 어떤 조건이나 이벤트가 주어졌을 때, 묵시적으로 컨트롤의 상태나 이벤트 핸들러를 호출하는 기능.
 - 트리거를 사용하면 변화에 따른 변화를 지정할 수 있다. 즉 element의 Property나 Data binding, event에서 변화가 발생할 때 element와 control이 어떤 반응을 할 수 있도록 만들 수 있다.
- Style에서 Setter와 Trigger의 차이
 - **Setter** : element가 처음 생성되는 시점에 Property를 설정한다.
 - **Trigger** : Property가 변경되는 시점에 Property를 설정한다.

Advanced WPF Programming

WPF Trigger 이해

- Trigger 종류
 - **Property Trigger** : Dependency Property가 변경될 때 실행된다.
 - **Event Trigger** : 라우티드 이벤트(Routed event)가 발생할 때 실행된다.
 - **Data Trigger** : Binding 문법으로 연결된 .NET Property 값이 변경되었을 때 실행된다.
 - **Multi Trigger** : 다수의 Property를 트리거 조건으로 걸고, 조건을 모두 만족하면 실행된다.

Advanced WPF Programming

WPF Trigger 이해

- Property Trigger 이해

```
<Window x:Class="WpfApp1.MainWindow"
```

```
...
```

```
Title="MainWindow" Height="300" Width="400">
```

```
<Grid>
```

```
<TextBlock Name="tblk_hello" Text="Hello, WPF world!!!" FontSize="20"
    HorizontalAlignment="Center" VerticalAlignment="Center">
```

```
<TextBlock.Style>
```

```
<Style TargetType="TextBlock">
```

```
<Setter Property="Foreground" Value="Green"> </Setter>
```

```
<Style.Triggers>
```

```
<Trigger Property="IsMouseOver" Value="True">
```

```
<Setter Property="Foreground" Value="Red"/>
```

```
<Setter Property="TextDecorations" Value="Underline"/>
```

```
</Trigger>
```

```
</Style.Triggers>
```

```
</Style>
```

```
</TextBlock.Style>
```

```
</TextBlock>
```

```
</Grid>
```

```
</Window>
```

Hello, WPF world



Hello, WPF world

Advanced WPF Programming

WPF Trigger 이해

- Property Trigger 이해

```
<Window x:Class="WpfApp1.MainWindow"
...
Title="MainWindow" Height="300" Width="400">
<Window.Resources>
  <Style x:Key="MyStyle">
    <Setter Property="Control.Foreground" Value="Red" />
    <Setter Property="TextBlock.Text" Value="Hello WPF" />
    <Style.Triggers>
      <Trigger Property="Control.IsMouseOver" Value="True">
        <Setter Property="Control.Foreground" Value="Blue"/>
        <Setter Property="TextBlock.Text" Value="Trigger 실행중..." />
      </Trigger>
    </Style.Triggers>
  </Style>
</Window.Resources>
<StackPanel>
  <Button Width="100" Height="70"
    Style="{StaticResource MyStyle}" Content="Trigger" />
  <TextBlock Style="{StaticResource MyStyle}"
    FontSize="20" HorizontalAlignment="Center" VerticalAlignment="Center"/>
</StackPanel>
</Window>
```

Advanced WPF Programming

WPF Trigger 이해

- Event Trigger 이해
 - 이벤트 트리거는 애니메이션이 필요할 때 사용된다.
 - 애니메이션을 사용하려면 스토리보드 객체를 생성해야 한다.
 - 스토리보드 생성 할 때 설정할 내용
 - StoryBoard의 타입,
 - Animation 타입,
 - TargetProperty,
 - Duration,
 - Target Property에 넣을 값(To)

Advanced WPF Programming

WPF Trigger 이해

- Event Trigger 이해

```
<Window x:Class="TestProject.MainWindow"
...
Title="MainWindow" Height="450" Width="585">
<Grid Margin="0,0,10,0">
  <Rectangle Name="rctback" HorizontalAlignment="Left" Height="100" Stroke="Black"
    VerticalAlignment="Top" Width="100" Margin="130,89,0,0" >
    <Rectangle.Style>
      <Style TargetType="Rectangle">
        <Setter Property="Fill" Value="YellowGreen"> </Setter>
        <Style.Triggers>
          <EventTrigger RoutedEvent="MouseEnter">
            <EventTrigger.Actions>
              <BeginStoryboard>
                <Storyboard>
                  <DoubleAnimation Duration="0:0:0.800" Storyboard.TargetProperty="Height" To="300" />
                  <DoubleAnimation Duration="0:0:0.800" Storyboard.TargetProperty="Width" To="300" />
                </Storyboard>
              </BeginStoryboard>
            </EventTrigger.Actions>
          </EventTrigger>
        </Style.Triggers>
      </Style>
    </Rectangle>
  </Grid>
</Window>
```


Advanced WPF Programming

WPF Trigger 이해

- Event Trigger 이해

```
<EventTrigger RoutedEvent="MouseLeave">
    <EventTrigger.Actions>
        <BeginStoryboard>
            <Storyboard>
                <DoubleAnimation Duration="0:0:0.800" Storyboard.TargetProperty="Height" To="100" />
                <DoubleAnimation Duration="0:0:0.800" Storyboard.TargetProperty="Width" To="100" />
            </Storyboard>
        </BeginStoryboard>
    </EventTrigger.Actions>
</EventTrigger>
```

```
</Style.Triggers>
```

```
</Style>
```

```
</Rectangle.Style>
```

```
</Rectangle>
```

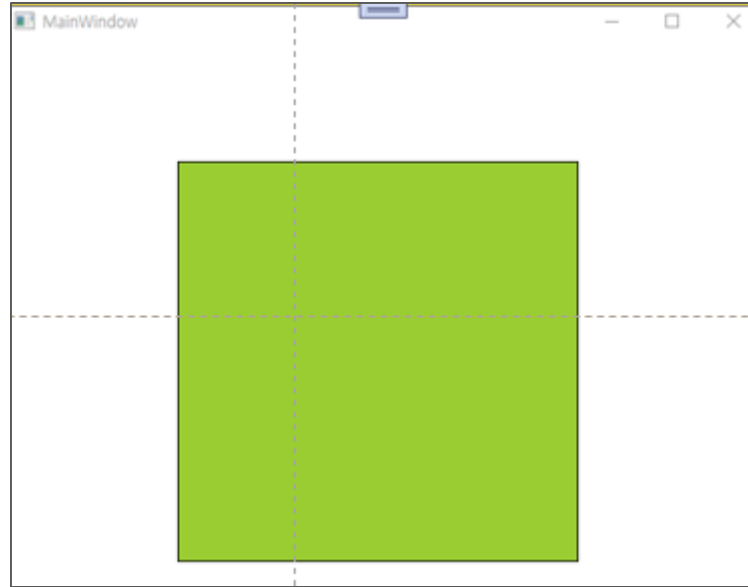
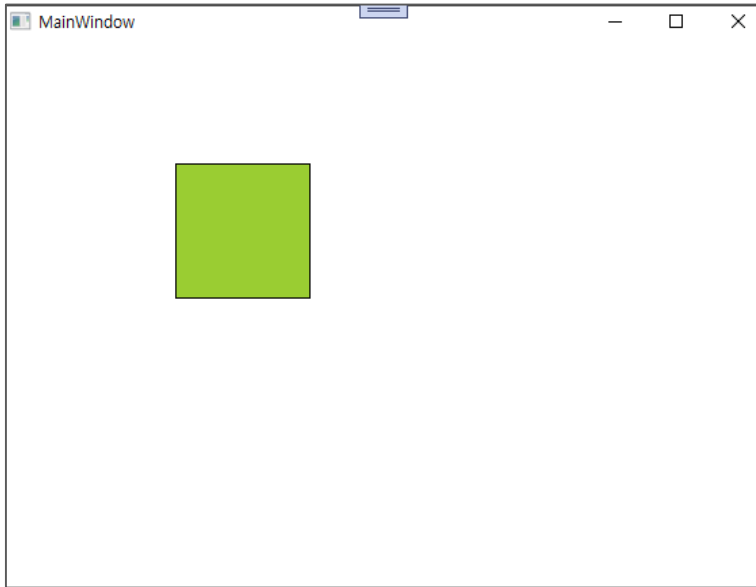
```
</Grid>
```

```
</Window>
```

Advanced WPF Programming

WPF Trigger 이해

- Event Trigger 이해



Advanced WPF Programming

WPF Trigger 이해

- Data Trigger 이해
 - DataTrigger 클래스는 Property Trigger의 Property를 binding으로 대신하는 것을 제외하고는 Trigger와 유사하며, binding은 다른 element를 참조한다.
 - DataTrigger는 binding 되는 값이 특정 값을 가질 때 Property를 설정할 수 있게 해준다.
 - Element로 표시하며 Trigger는 의존 속성이 아닌 속성에 사용된다.
 - **Model View ViewModel(M-V-VM)** 디자인 패턴을 사용하여 데이터 바인딩을 사용하는 경우 이상적이다.

Advanced WPF Programming

WPF Trigger 이해

- Data Trigger 이해

```
<Window x:Class="WpfApp3.MainWindow"
```

```
...
```

```
<Window.Resources>
```

```
<Style x:Key="MyStyle" TargetType="TextBlock">
```

```
<Setter Property="Visibility" Value="Visible"/>
```

```
<Style.Triggers>
```

```
<DataTrigger Binding="{Binding ElementName=cb1, Path=IsChecked}" Value="True">
```

```
<Setter Property="Visibility" Value="Hidden"/>
```

```
</DataTrigger>
```

```
</Style.Triggers>
```

```
</Style>
```

```
</Window.Resources>
```

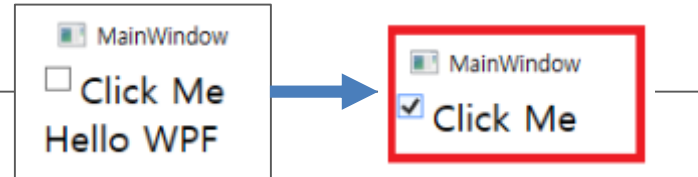
```
<StackPanel>
```

```
<CheckBox Name="cb1" Content="Click Me" FontSize="20"/>
```

```
<TextBlock Text="Hello WPF" FontSize="20" Style="{StaticResource MyStyle}"/>
```

```
</StackPanel>
```

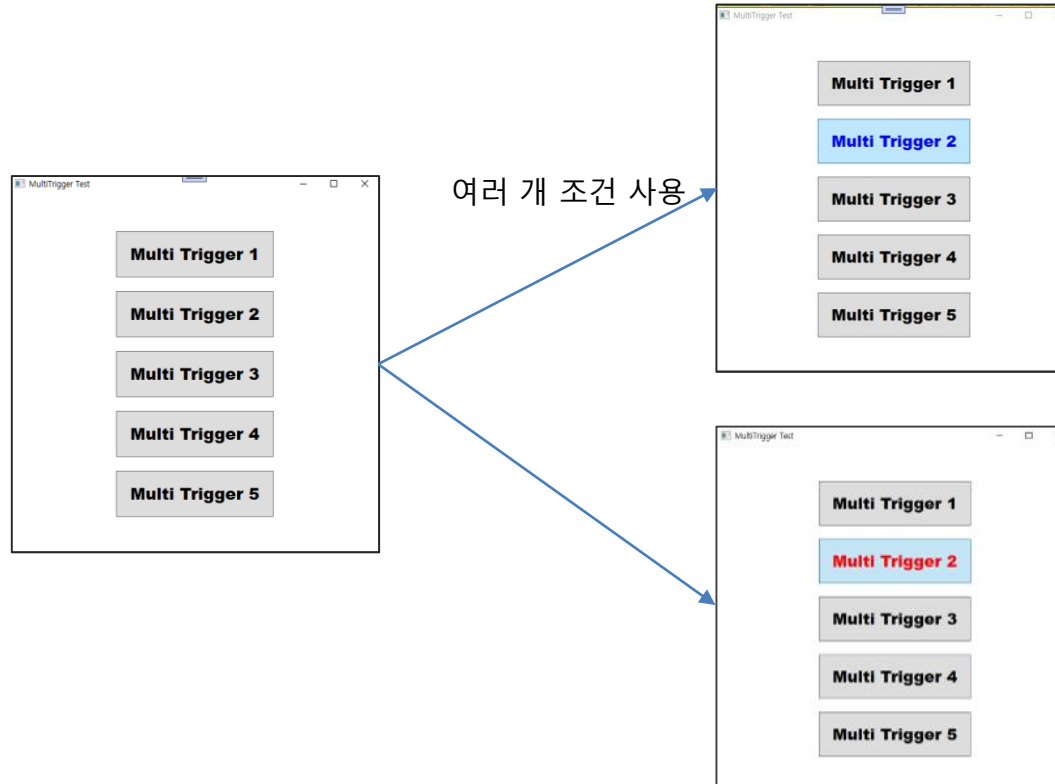
```
</Window>
```



Advanced WPF Programming

WPF Trigger 이해

- Multi Trigger 이해
 - 여러 개 조건을 and로 설정할 수 있는 trigger



Advanced WPF

WPF Trigger 이해

- Multi Trigger 이해

```
<Window
...
FontFamily="Arial Black"
FontSize="16"
Title="MultiTrigger Test">
  <StackPanel HorizontalAlignment="Center" VerticalAlignment="Center">
    <StackPanel.Resources>
      <Style x:Key="multi">
        <Setter Property="FrameworkElement.HorizontalAlignment" Value="Center" />
        <Setter Property="FrameworkElement.Margin" Value="10" />
        <Setter Property="Control.Padding" Value="20 10 20 10" />
        <Setter Property="Control.FontSize" Value="24" />
        <Style.Triggers>
          <Trigger Property="ButtonBase.IsPressed" Value="True">
            <Setter Property="Control.Foreground" Value="Red" />
          </Trigger>
          <MultiTrigger>
            <MultiTrigger.Conditions>
              <Condition Property="UIElement.IsMouseOver" Value="True" />
              <Condition Property="ButtonBase.IsPressed" Value="False" />
            </MultiTrigger.Conditions>
            <Setter Property="Control.FontStyle" Value="Italic" />
            <Setter Property="Control.Foreground" Value="Blue" />
          </MultiTrigger>
        </Style.Triggers>
      </Style>
    </StackPanel.Resources>

    <Button Style="{StaticResource multi}" Content="Multi Trigger 1" Height="67"/>
    <Button Style="{StaticResource multi}" Content="Multi Trigger 2" Height="67"/>
    <Button Style="{StaticResource multi}" Content="Multi Trigger 3" Height="67"/>
    <Button Style="{StaticResource multi}" Content="Multi Trigger 4" Height="67"/>
    <Button Style="{StaticResource multi}" Content="Multi Trigger 5" Height="67"/>
  </StackPanel>
</Window>
```

[C# 기반]

Advanced WPF Programming

WPF COMMAND

Advanced WPF Programming

WPF Command 이해

- Command란?
 - WPF의 Input 매커니즘의 하나인 Command
 - .xaml 파일에서 버튼, 체크박스 버튼 등의 컨트롤 동작을 이벤트로 처리할 때 사용하는 인터페이스인 ICommand 를 상속받은 클래스에서 이벤트를 처리하는 방법
 - ICommand에서 Execute, CanExecute 메소드 제공, CanExecuteChanged 이벤트 제공
 - Execute : 실제 처리해야하는 작업을 작성하는 메소드
 - CanExecute : Execute 메소드의 코드를 실행할 지의 여부 결정
 - CanExecuteChanged : ICommand 구현체에서 CanExecuteChanged 이벤트를 CommandManager의 RequerySuggested 이벤트에 연결하여 사용.
 - 여러 개의 버튼에서 하나의 Command를 공유하는 사용할 수 있도록 한다.
 - Event 처리 방법을 MVVM 패턴에서 사용하는 방식으로 MVVM 패턴 구조로 설계

[command 참고]

<https://learn.microsoft.com/ko-kr/dotnet/desktop/wpf/advanced/commanding-overview?view=netframeworkdesktop-4.8&viewFallbackFrom=netdesktop-6.0>

Advanced WPF Programming

WPF Command 이해

- Command를 사용하는 이유
 - Command 발생시키는 Object와 Command 실행하는 Login을 분리
- Action 실행이 가능한지를 확인하기위해 사용
 - CanExcute, CanExcuteChanged
- Command를 구성하는 main concept
 - Command : 실행할 액션
 - Command Source : Command를 호출하는 Object
 - Command Target : Action 목적지
 - Command Binding : Command와 Command Login 연결

Advanced WPF Programming

WPF Command 이해

```
<StackPanel>
  <Menu>
    <MenuItem Command="ApplicationCommands.Paste"/>
  </Menu>
  <TextBox/>
</StackPanel>
```

- command : paste
- command source : menuItem
- command target : textbox
- command binding : textbox support

Advanced WPF Programming

WPF Command 이해

- Command
 - WPF IComamnd interface를 구현하여 만든다.
 - ICommand는 다음 2개 메서드와 1개 이벤트 인터페이스를 갖고 있다.
 - Execute
 - CanExecute
 - CanExecuteChanged
 - Execute는 command와 관련한 action을 수행한다.
CanExecute는 현재 command target에 command 수행할 수 있는지 조사하여 결과를 리턴한다.
CanExecuteChanged는 변화가 있는 command source를 탐지하고 comamnd가 아직 실행되기 전에 불린다.
 - RoutedCommand 자체는 application logic을 갖고 있지 않다. 대신 routed event를 발생시키고 element tree를 따라 CommandddBinding을 만날때까지 이 이벤트를 전파한다.
CommandBinding은 이벤트 핸들러를 갖고 있다.

Advanced WPF Programming

WPF Command 이해

- Comamnd Sources
 - command를 발생시키는 object이다. object는 일반적으로 ICommandSource interface를 구현한다. ICommandSource interface는 다음과 같다.
 - Command
object가 발생시킬 command
 - CommandTarget
command가 작용할 object
CommandTarget은 오직 ICommand가 RoutedCommand일 때만 작동함.
만약 command가 RoutedCommand가 아니면 CommandTarget은 무시된다.
CommandTarget을 적용하지 않으면, keyboard focus를 가진 element가 ComamndTarget이 된다.
 - CommandParameter
user-defined data type. Command 구현한 핸들러에게 넘겨준다.
 - InputGesture는 comamnd와 어떤 동작을 연결한다.
InputGesture는 command source에 적용한다. wpf inputgesture에는 다음 두 가지가 있다.
 - KeyGesture
 - MouseGesture

```
<Window.InputBindings>
  <KeyBinding Key="B"
              Modifiers="Control"
              Command="ApplicationCommands.Open" />
</Window.InputBindings>
```

Advanced WPF Programming

WPF Command 이해

- CommandBinding
 - CommandBinding은 command와 command를 구현하는 이벤트 핸들러를 연결한다. 다음의 구성요소를 갖고 있다.
 - Command Property
CommandBinding과 연결할 Command
 - PreviewExecuted event
Command logic을 구현
 - Executed event
Command logic을 구현
 - PreviewExecuted event Command가 동작할 수 있는지 결정
 - CanExecute event
Command가 동작할 수 있는지 결정

Advanced WPF Programming

WPF Command 이해

- Command Target
 - command가 동작할 element. ICommandSource의 CommandTarget은 ICommand가 RoutedCommand 일 때만 사용가능하다.

```
<Window.CommandBindings>
<CommandBinding Command="ApplicationCommands.Open"
                Executed="OpenCmdExecuted"
                CanExecute="OpenCmdCanExecute"/> </Window.CommandBindings>

<Grid>
  <StackPanel>
    <Button Name="pasteBtn" Width="50"
            Command="ApplicationCommands.Open">paste</Button>
  </StackPanel>
</Grid>
```

Advanced WPF Programming

WPF Command 이해

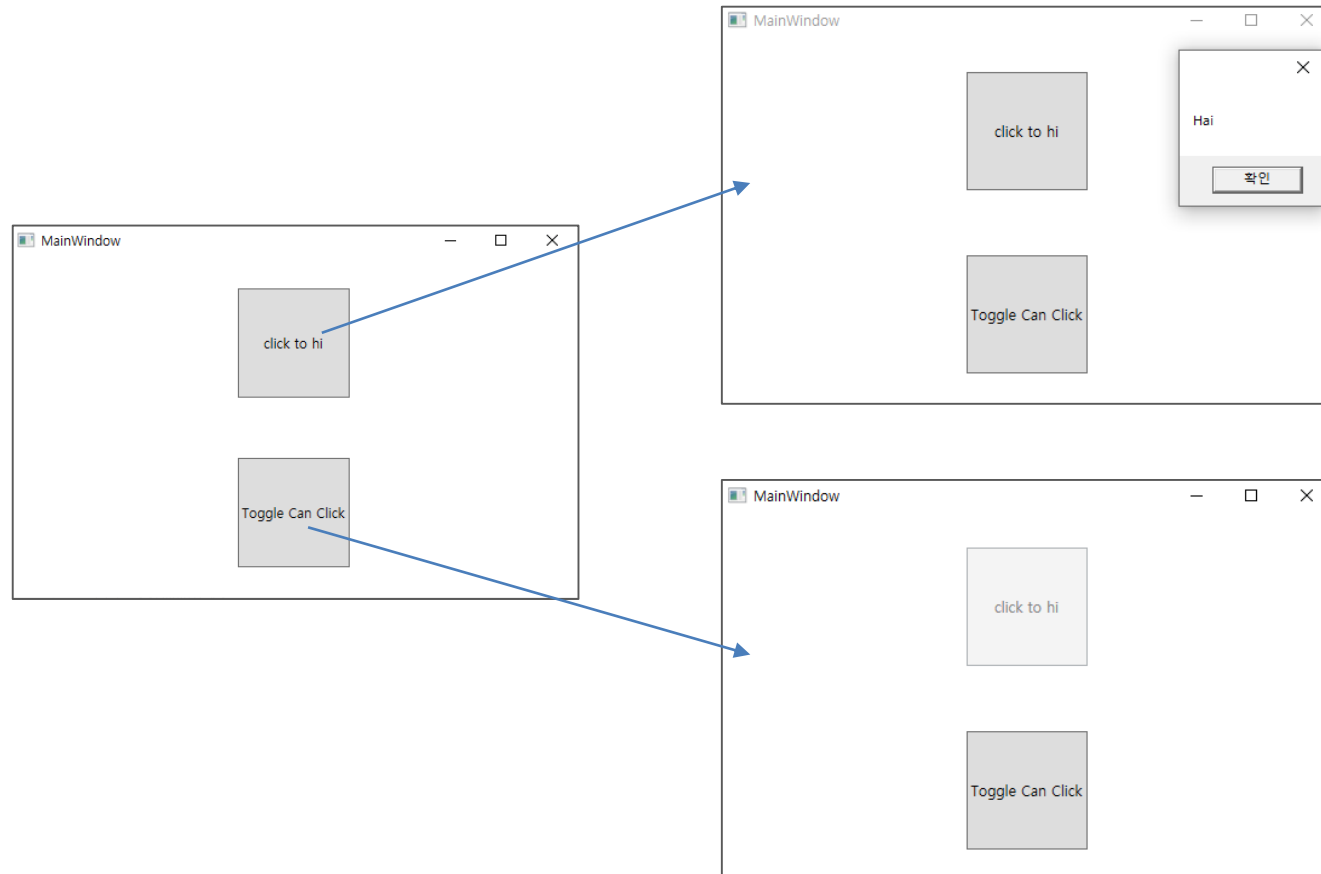
```
private void OpenCmdExecuted(object sender, ExecutedRoutedEventArgs e)
{
    string command, targetobj;
    command = ((RoutedCommand)e.Command).Name;
    targetobj = ((FrameworkElement)e.Source).Name;
    MessageBox.Show("The " + command +
                    " command has benn invoked on target object " + targetobj);
}

private void OpenCmdCanExecute(object sender, CanExecuteRoutedEventArgs e)
{
    e.CanExecute = true;
}
```

Advanced WPF Programming

WPF Command 이해

- Command를 이용한 이벤트 처리



Advanced WPF Programming

WPF Command 이해

- MainWindowView.xaml

```
<Window x:Class="WpfExample.MainWindowView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="MainWindow" Height="350" Width="525"
    xmlns:local="clr-namespace:WpfExample">

    <Window.DataContext>
        <local:MainWindowViewModel/>
    </Window.DataContext>

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition/>
        </Grid.RowDefinitions>

        <Button Grid.Row="0" Command="{Binding HiButtonCommand}" CommandParameter="Hai" Content="{Binding HiButtonContent}"
            Width="100"
            Height="100" />

        <Button Grid.Row="1" Content="Toggle Can Click" Command="{Binding ToggleExecuteCommand}" Width="100" Height="100"/>
    </Grid>

</Window>
```

Advanced WPF Programming

WPF Command 이해

- MainWindowViewModel.cs

```
class MainWindowViewModel
{
    private ICommand hiButtonCommand;

    private ICommand toggleExecuteCommand { get; set; }

    private bool canExecute = true;

    public string HiButtonContent
    {
        get { return "click to hi"; }
    }

    public bool CanExecute
    {
        get { return this.canExecute; }

        set
        {
            if (this.canExecute == value) { return; }
            this.canExecute = value;
        }
    }
}
```

```
public ICommand ToggleExecuteCommand
{
    get { return toggleExecuteCommand; }
    set { toggleExecuteCommand = value; }
}

public ICommand HiButtonCommand
{
    get { return hiButtonCommand; }
    set { hiButtonCommand = value; }
}

public MainWindowViewModel()
{
    HiButtonCommand = new RelayCommand(ShowMessage, param => this.canExecute);
    toggleExecuteCommand = new RelayCommand(ChangeCanExecute);
}

public void ShowMessage(object obj)
{
    MessageBox.Show(obj.ToString());
}

public void ChangeCanExecute(object obj)
{
    canExecute = !canExecute;
}
}
```

Advanced WPF Programming

WPF Command 이해

- RelayCommand.cs

```
public class RelayCommand : ICommand
{
    private Action<object> execute;
    private Predicate<object> canExecute;
    public RelayCommand(Action<object> execute)
        : this(execute, DefaultCanExecute) { }

    public RelayCommand(Action<object> execute, Predicate<object> canExecute)
    {
        if (execute == null)
        {
            throw new ArgumentNullException("execute");
        }

        if (canExecute == null)
        {
            throw new ArgumentNullException("canExecute");
        }

        this.execute = execute;
        this.canExecute = canExecute;
    }
}
```

Advanced WPF Programming

WPF Command 이해

- RelayCommand.cs 계속

```
public event EventHandler CanExecuteChanged
{
    add
    {
        CommandManager.RequerySuggested += value;
        this.CanExecuteChangedInternal += value;
    }

    remove
    {
        CommandManager.RequerySuggested -= value;
        this.CanExecuteChangedInternal -= value;
    }
}

private event EventHandler CanExecuteChangedInternal;

public bool CanExecute(object parameter)
{
    return this.canExecute != null && this.canExecute(parameter);
}

public void Execute(object parameter)
{
    this.execute(parameter);
}
```

Advanced WPF Programming

WPF Command 이해

- RelayCommand.cs 계속

```
public void OnCanExecuteChanged()
{
    EventHandler handler = this.CanExecuteChangedInternal;

    if (handler != null)
    {
        //DispatcherHelper.BeginInvokeOnUIThread(() => handler.Invoke(this, EventArgs.Empty));
        handler.Invoke(this, EventArgs.Empty);
    }
}

public void Destroy()
{
    this.canExecute = _ => false;
    this.execute = _ => { return; };
}

private static bool DefaultCanExecute(object parameter)
{
    return true;
}
}
```

[C# 기반]

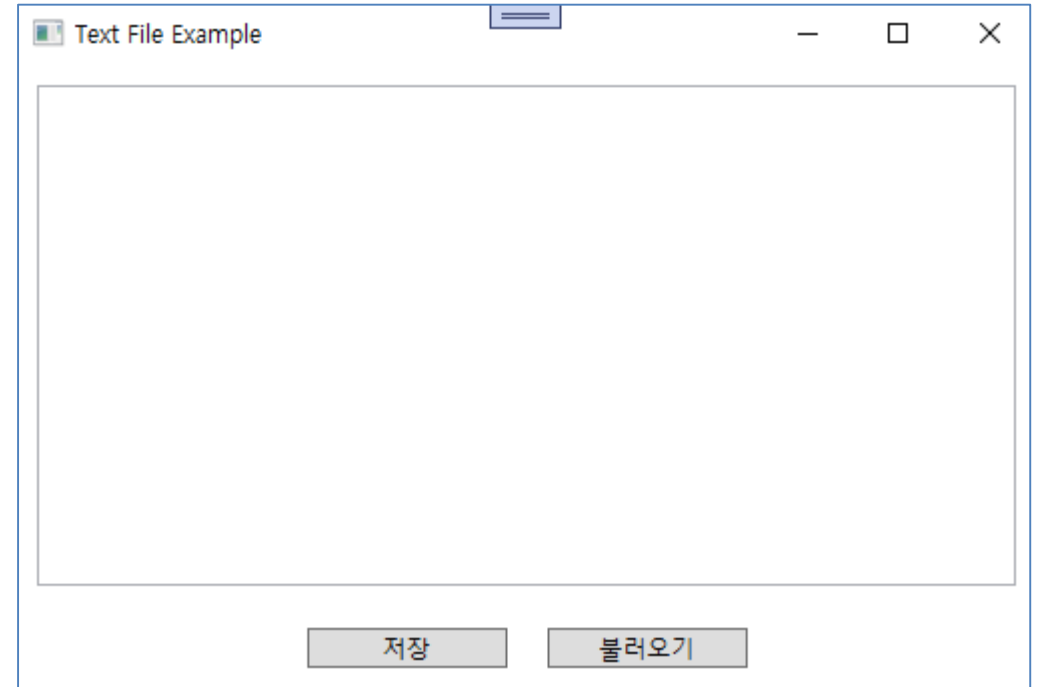
WPF GUI Programming 심화

WPF 파일 연동

Advanced WPF Programming

WPF TXT 파일 연동

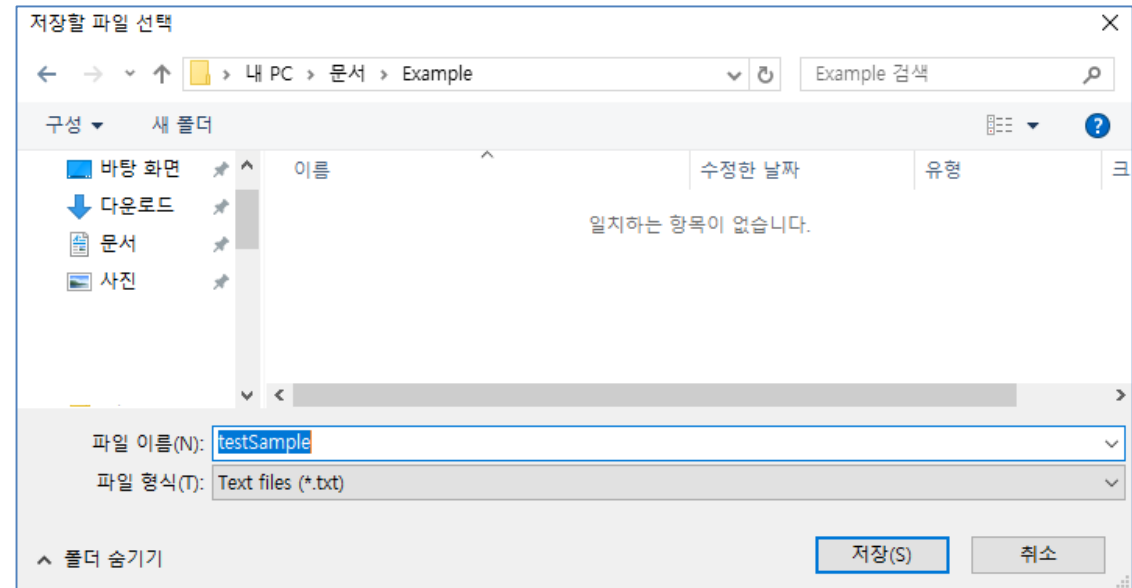
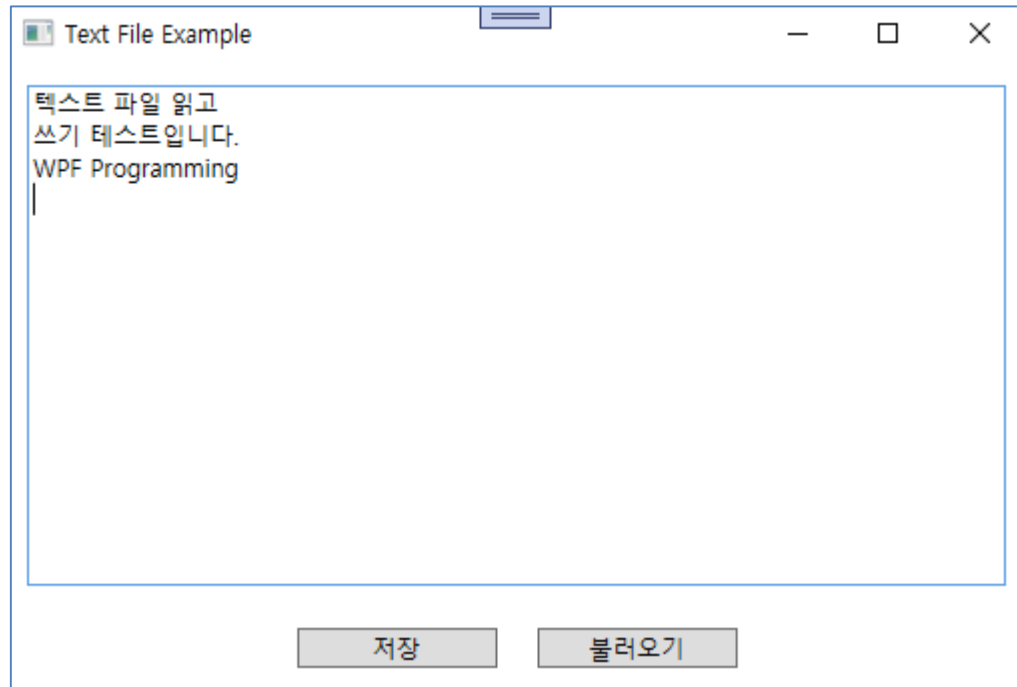
- 텍스트 파일 다루기 위해 System.IO의 기능을 이용
- 기능
 - TextBox에 입력된 내용을 파일로 저장하고,
 - 저장된 파일을 읽어와서 TextBox에 표시
- 주요 클래스
 - File
 - SaveFileDialog
 - OpenFileDialog



Advanced WPF Programming

WPF TXT 파일 연동

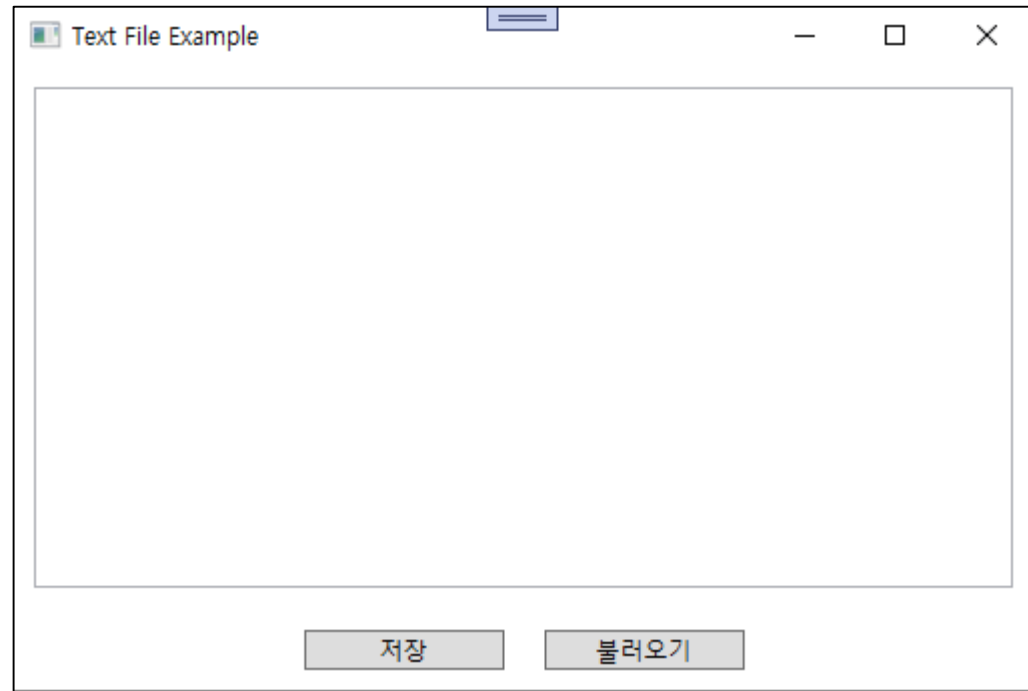
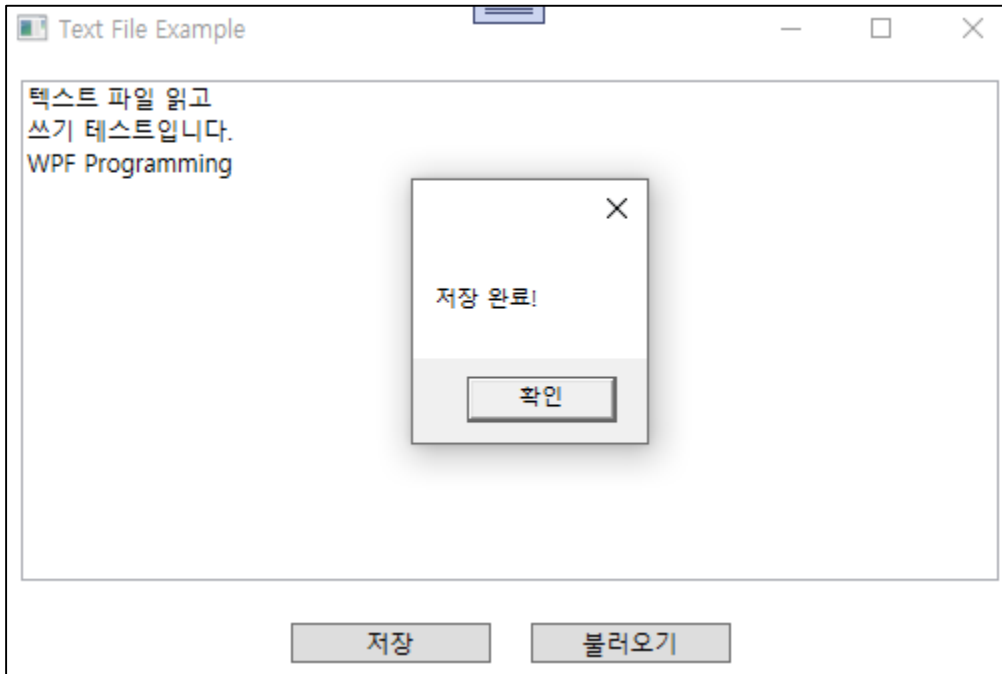
- 텍스트 입력 후 저장 버튼을 선택할 때



Advanced WPF Programming

WPF TXT 파일 연동

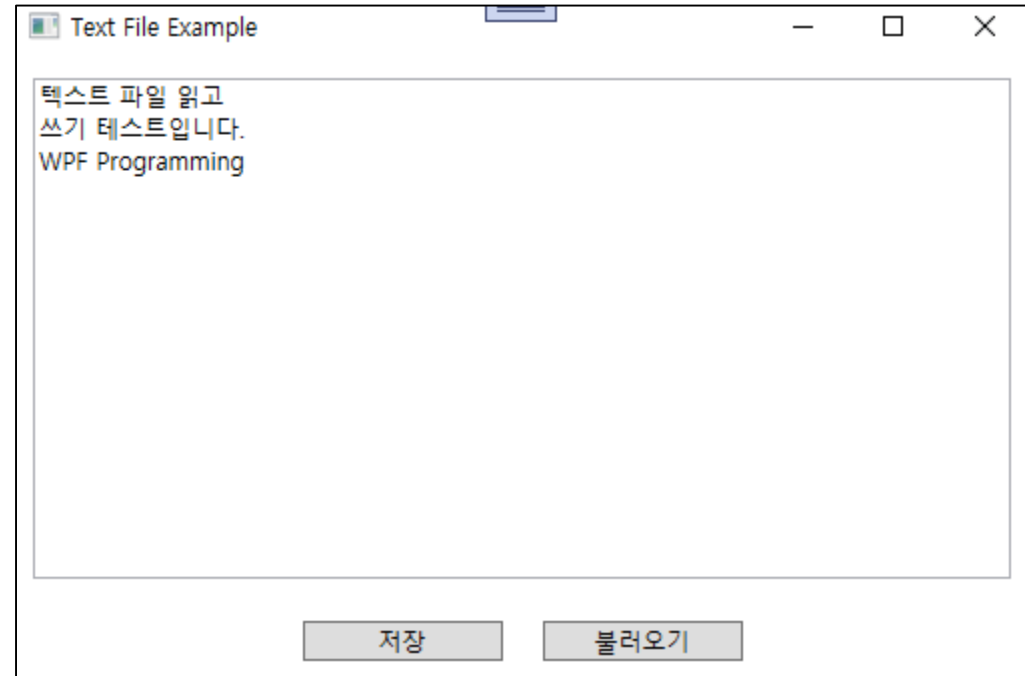
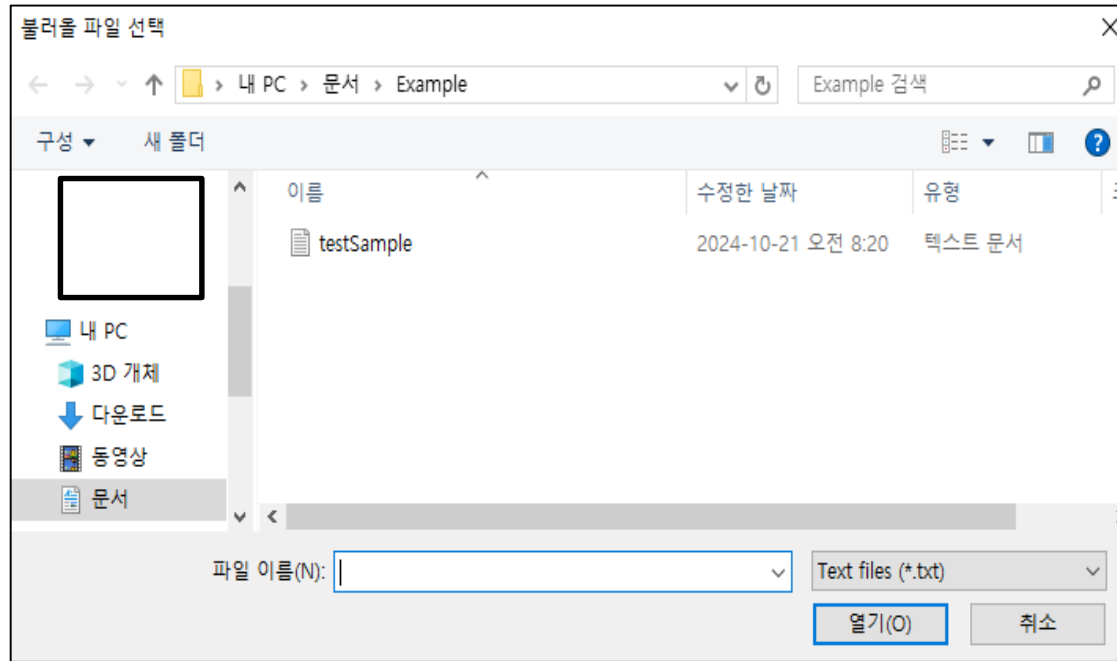
- 저장 확인과 저장 후 텍스트 입력 부분은 초기화



Advanced WPF Programming

WPF TXT 파일 연동

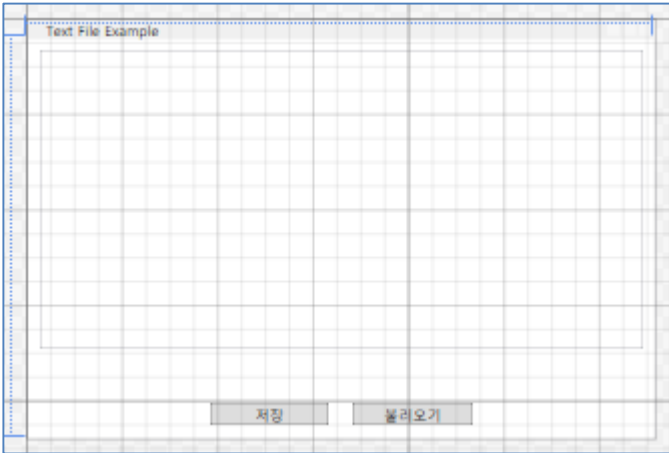
- 파일을 선택하고 불러오기



Advanced WPF Programming

WPF TXT 파일 연동

- 텍스트 입력 UI



```
<Grid>
  <TextBox x:Name="InputTextBox"
    VerticalAlignment="Top"
    Height="250"
    AcceptsReturn="True"
    TextWrapping="Wrap"
    Margin="10" />
  <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
    <Button Content="저장"
      VerticalAlignment="Bottom"
      Width="100"
      Margin="10, 10, 10, 10"
      Click="SaveButton_Click" />
    <Button Content="불러오기"
      VerticalAlignment="Bottom"
      Width="100"
      Margin="10, 10, 10, 10"
      Click="LoadButton_Click" />
  </StackPanel>
</Grid>
```

Advanced WPF Programming

WPF TXT 파일 연동

```
private void SaveButton_Click(object sender, RoutedEventArgs e)
{
    SaveFileDialog saveFileDialog = new SaveFileDialog
    {
        Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*",
        Title = "저장할 파일 선택"
    };

    if (saveFileDialog.ShowDialog() == true)
    {
        try
        {
            File.WriteAllText(saveFileDialog.FileName, InputTextBox.Text);
            MessageBox.Show("저장 완료!");
            InputTextBox.Text = "";
        }
        catch (Exception ex)
        {
            MessageBox.Show($"오류 발생: {ex.Message}");
        }
    }
}
```

Advanced WPF Programming

WPF TXT 파일 연동

```
private void LoadButton_Click(object sender, RoutedEventArgs e)
{
    OpenFileDialog openFileDialog = new OpenFileDialog
    {
        Filter = "Text files (*.txt)|*.txt|All files (*.*)|*.*",
        Title = "불러올 파일 선택"
    };

    if (openFileDialog.ShowDialog() == true)
    {
        try
        {
            InputTextBox.Text = File.ReadAllText(openFileDialog.FileName);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"오류 발생: {ex.Message}");
        }
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동

- Excel 파일을 읽고 쓸 수 있는 라이브러리 종류

라이브러리	파일 포맷 지원	주요 기능	무료 사용 조건
EPPlus	.xlsx	Excel 파일 읽기/쓰기, 차트, 수식 지원 등	비상업적 용도로 무료
ClosedXML	.xlsx	Excel 파일 읽기/쓰기, 수식, 스타일링 등	상업적/비상업적 모두 무료
NPOI	.xlsx, .xls	Excel, Word, PowerPoint 파일 읽기/쓰기	상업적/비상업적 모두 무료
ExcelDataReader	.xlsx, .xls	Excel 파일 읽기(읽기전용)	상업적/비상업적 모두 무료
Syncfusion XlsIO	.xls, .xlsx	대규모 데이터 처리, 차트, PDF 내보내기	상용 라이선스 필요 (비상업적 무료)
DevExpress Spreadsheet	.xls, .xlsx,	피벗 테이블, 차트, 수식, 대규모 데이터 처리	상용 라이선스 필요
IronXL	.xls, .xlsx	빠르고 직관적인 API, 수식 계산, 차트 지원	상용 라이선스 필요
SpreadsheetGear	.xls, .xlsx	수식 계산, 차트, 대규모 데이터 처리	상용 라이선스 필요
Telerik Spreadsheet Processing	.xls, .xlsx	피벗 테이블, 차트, 수식, 다양한 UI 플랫폼 지원	상용 라이선스 필요

Advanced WPF Programming


WPF Excel 파일 연동


- ClosedXML 이용하기


NuGet - 솔루션 MainWindow.xaml MainWindow.xaml.cs


찾아보기 설치됨 업데이트 통합

ClosedXML x ↻ ☒ 시험판 포함

ClosedXML  작성자: Jan Havlíček, Francois Botha, Aleksei Pankratev, Manuel de Lec 0.104.1
See release notes <https://github.com/ClosedXML/ClosedXML/releases/tag/0.104.1>
ClosedXML is a .NET library for reading, manipulating and writing Excel 2007+ (.xlsx, .x...

ClosedXML.Report  작성자: Alexey Rozhkov, Alexey Pankratev, 2.87M개 다운로드 0.2.10
ClosedXML.Report is a tool for report generation and data analysis in .NET applications through the use of Microsoft Excel. ClosedXML.Report is a .NET-library for report generat...

ClosedXML.Extensions.Mvc  작성자: Francois Botha, 755K개 다운로드 0.4.0
MVC extensions for ClosedXML

ClosedXML.Signed  작성자: Francois Botha, Aleksei Pankratev, Manuel de Leon, 0.95.4
ClosedXML is a .NET library for reading, manipulating and writing Excel 2007+ (.xlsx, ...
이 패키지 버전은 더 이상 사용하지 않습니다

각 패키지는 해당 소유자에 의해 사용이 허가되었습니다. NuGet은 타사 패키지에 대해 책임을 지지 않으며 라이선스를 부여하지도 않습니다.

☐ 다시 표시하지 않음(D)


변경 내용 미리 보기

Visual Studio에서 이 솔루션을 변경하려고 합니다. 복사(P)

System.IO.FileSystem.Primitives.4.3.0
System.IO.FileSystem.4.3.0
System.IO.Packaging.8.0.0
System.Linq.4.3.0
System.Linq.Expressions.4.3.0
System.Net.Primitives.4.3.0
System.Net.Sockets.4.3.0
System.Numerics.Vectors.4.5.0
System.ObjectModel.4.3.0
System.Reflection.4.3.0
System.Reflection.Extensions.4.3.0
System.Reflection.Primitives.4.3.0
System.Resources.ResourceManager.4.3.0
System.Runtime.4.3.0
System.Runtime.CompilerServices.Unsafe.4.7.0
System.Memory.4.5.4
ClosedXML.Parser.1.2.0
SixLabors.Fonts.1.0.0
System.Runtime.Extensions.4.3.0
System.Runtime.Handles.4.3.0
System.Runtime.InteropServices.4.3.0
System.Runtime.InteropServices.RuntimeInformation.4.3.0
System.Runtime.Numerics.4.3.0
System.Security.Cryptography.Encoding.4.3.0
System.Security.Cryptography.Primitives.4.3.0
System.Security.Cryptography.Algorithms.4.3.0
System.Security.Cryptography.X509Certificates.4.3.0
System.Net.Http.4.3.0
System.Text.Encoding.4.3.0
System.Text.Encoding.Extensions.4.3.0
System.Text.RegularExpressions.4.3.0
System.Threading.4.3.0
System.Threading.Tasks.4.3.0
System.Threading.Timer.4.3.0
System.Xml.ReaderWriter.4.3.0
System.Xml.XDocument.4.3.0
NETStandard.Library.1.6.1
RBush.3.2.0
ClosedXML.0.104.1

설치됨: 설치되지 않음

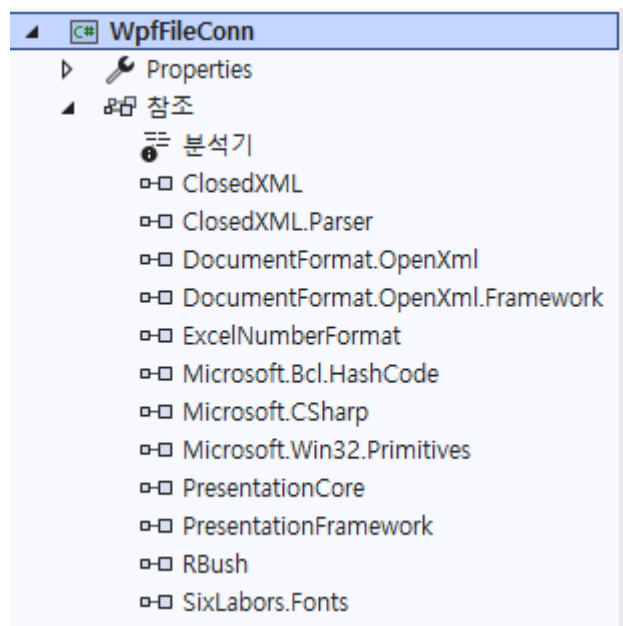
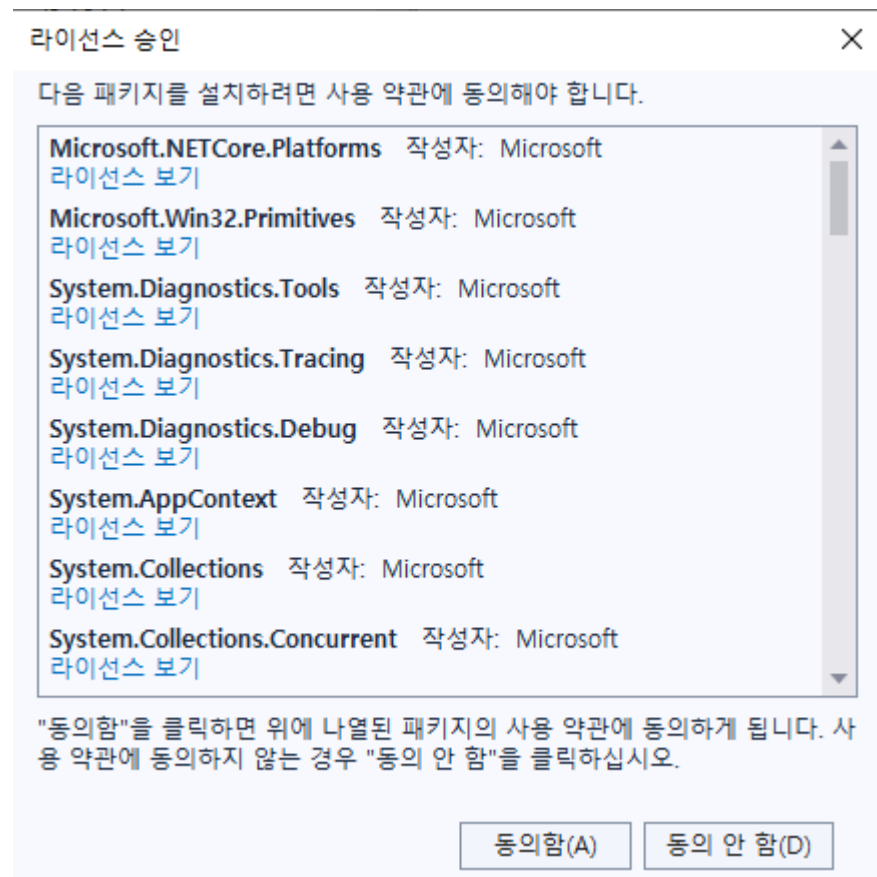
버전: 안정적인 최신 버전 0.104.1

 패키지 원본 매핑이 꺼져 있습니다

☐ 다시 표시하지 않음(D)

Advanced WPF Programming

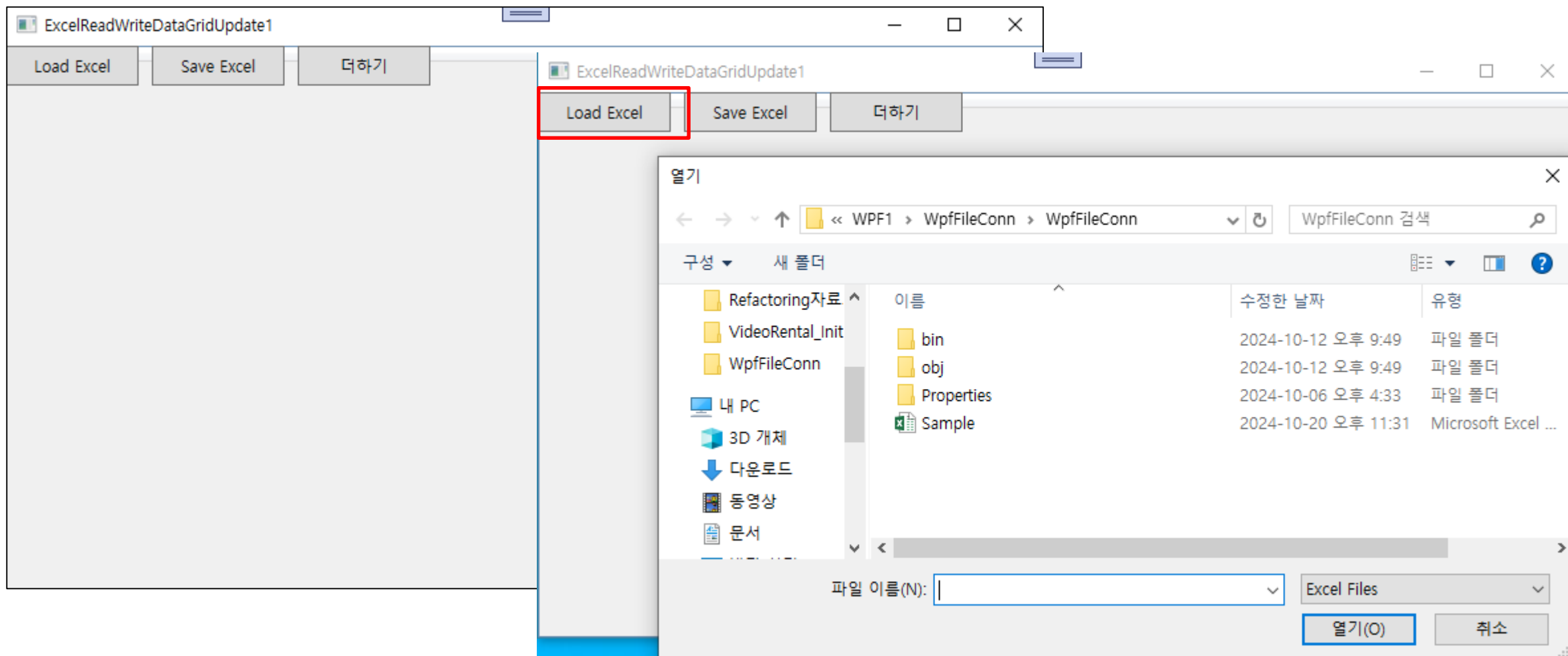
WPF Excel 파일 연동



Advanced WPF Programming

WPF Excel 파일 연동

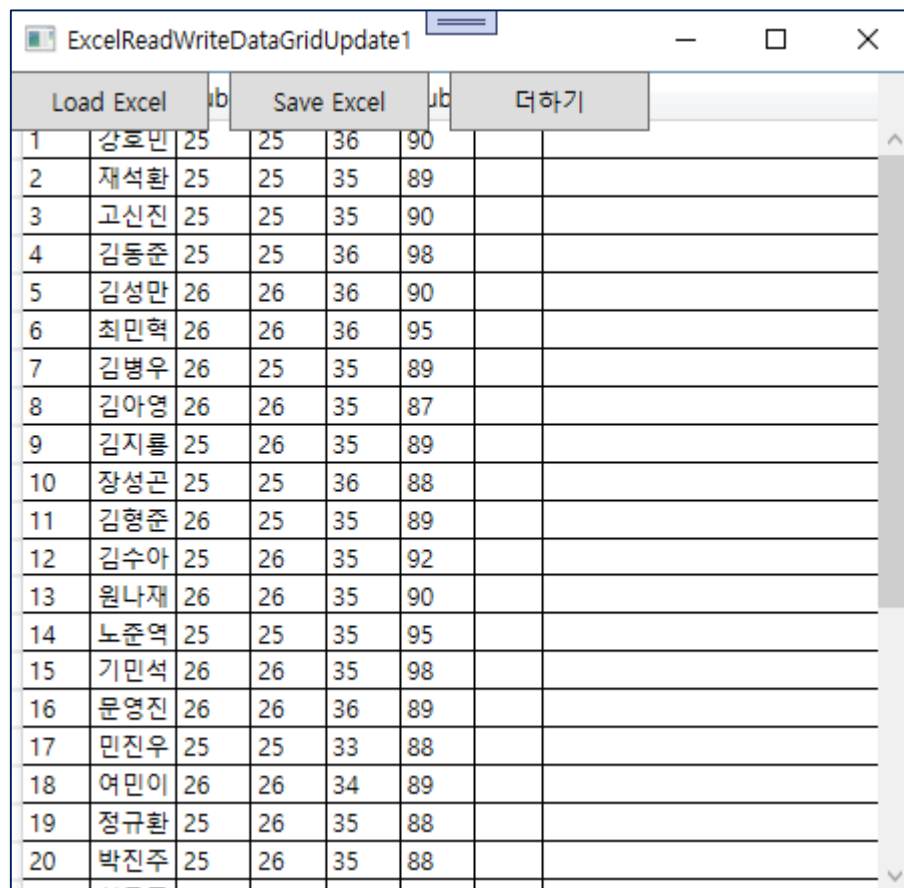
- 샘플 UI



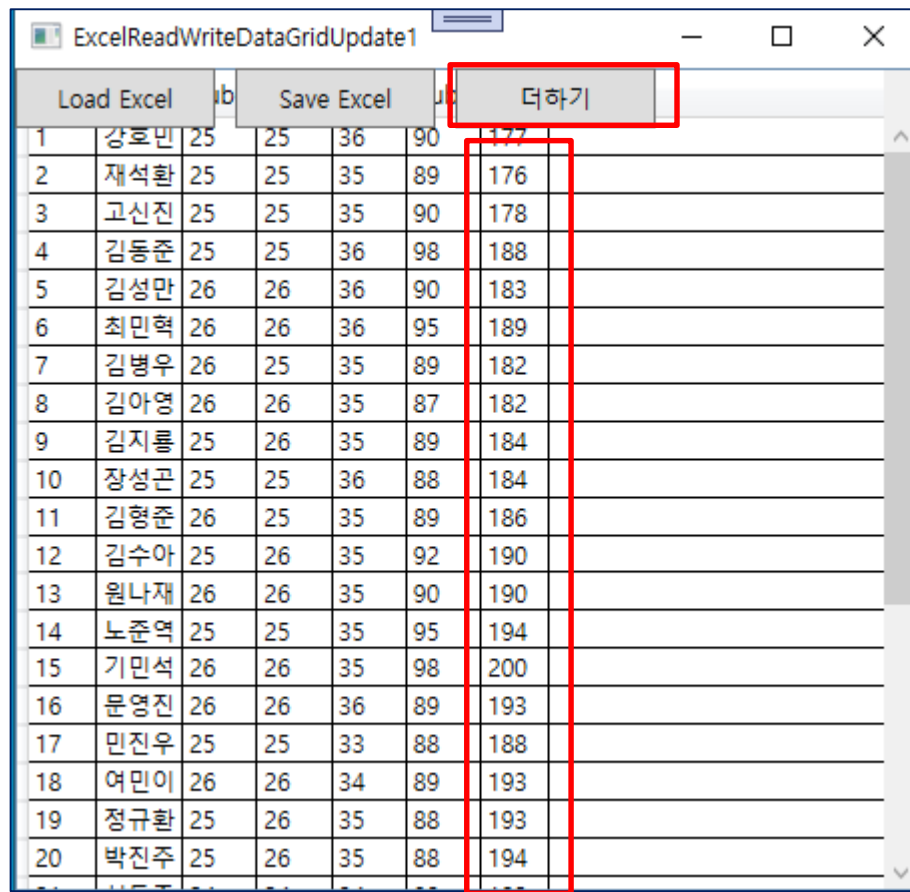
Advanced WPF Programming

WPF Excel 파일 연동

- 파일 로드, 내용 변경 및 저장 기능



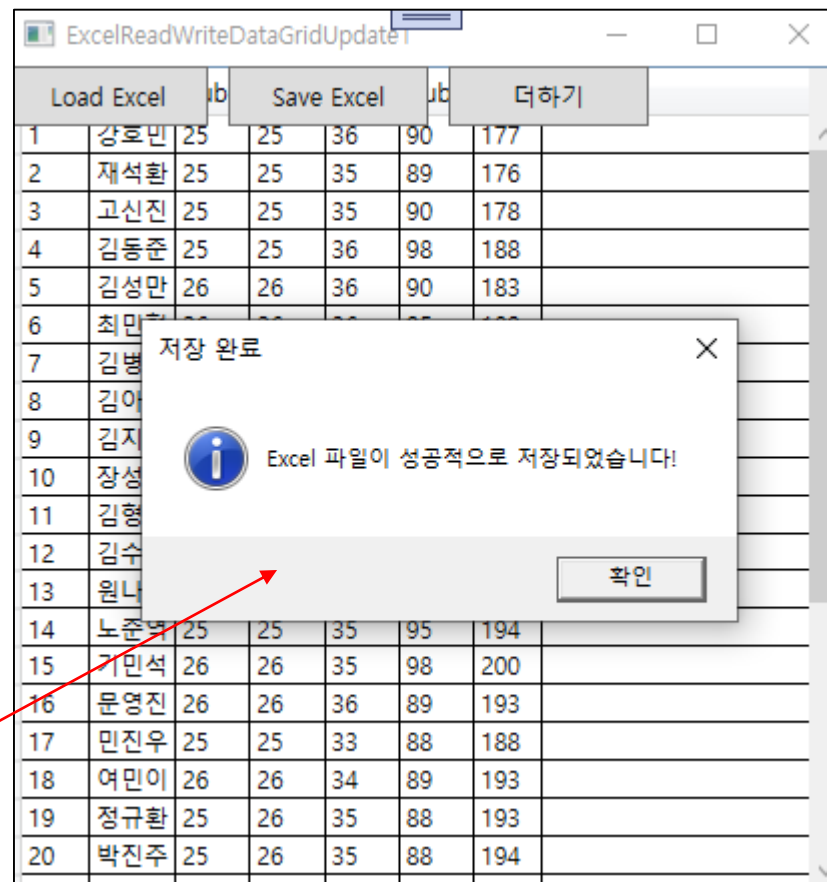
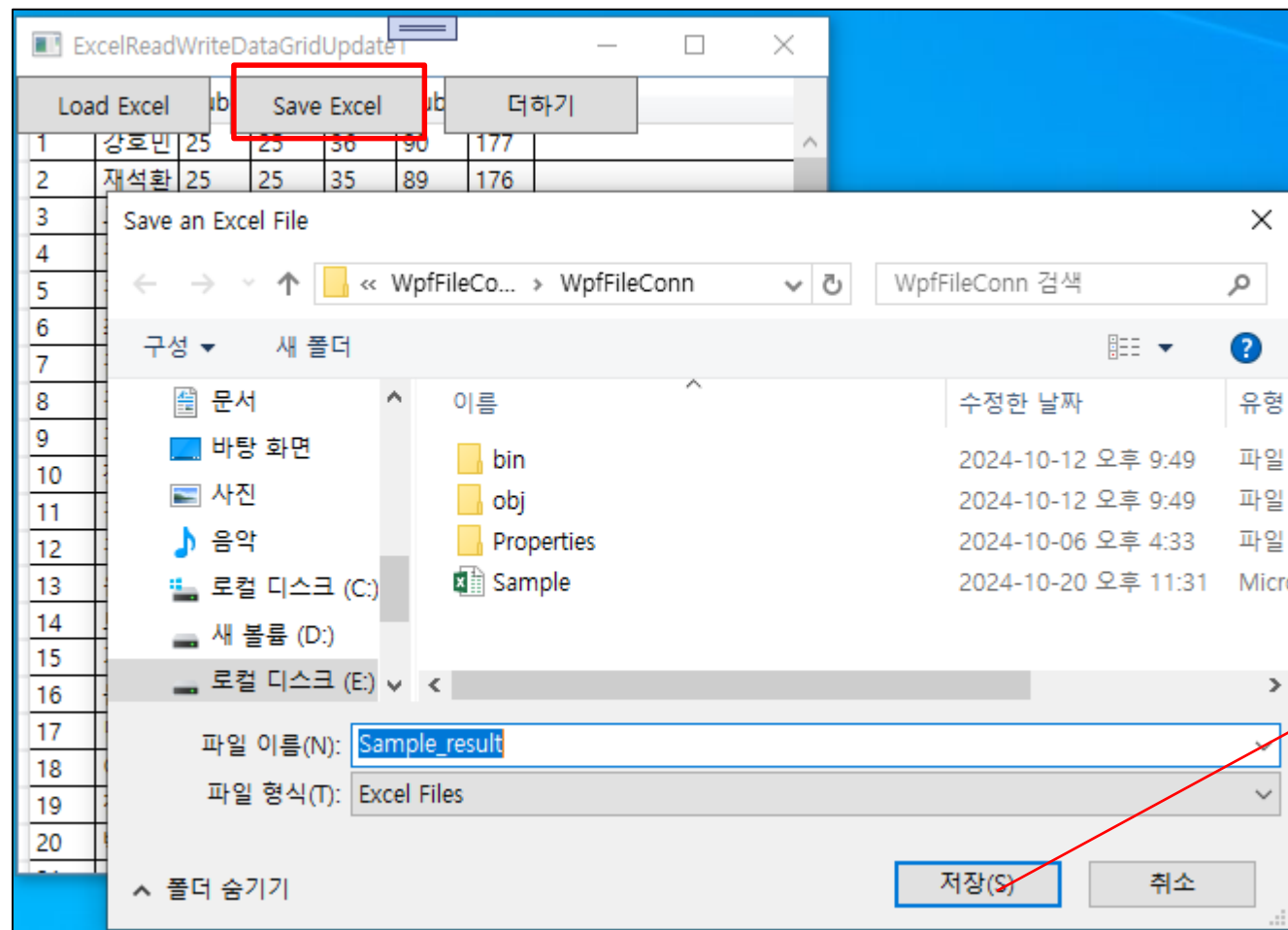
	Load Excel	Save Excel	더하기
1	강호민	25 25 36 90	
2	재석환	25 25 35 89	
3	고신진	25 25 35 90	
4	김동준	25 25 36 98	
5	김성만	26 26 36 90	
6	최민혁	26 26 36 95	
7	김병우	26 25 35 89	
8	김아영	26 26 35 87	
9	김지룡	25 26 35 89	
10	장성곤	25 25 36 88	
11	김형준	26 25 35 89	
12	김수아	25 26 35 92	
13	원나재	26 26 35 90	
14	노준역	25 25 35 95	
15	기민석	26 26 35 98	
16	문영진	26 26 36 89	
17	민진우	25 25 33 88	
18	여민이	26 26 34 89	
19	정규환	25 26 35 88	
20	박진주	25 26 35 88	



	Load Excel	Save Excel	더하기
1	강호민	25 25 36 90	177
2	재석환	25 25 35 89	176
3	고신진	25 25 35 90	178
4	김동준	25 25 36 98	188
5	김성만	26 26 36 90	183
6	최민혁	26 26 36 95	189
7	김병우	26 25 35 89	182
8	김아영	26 26 35 87	182
9	김지룡	25 26 35 89	184
10	장성곤	25 25 36 88	184
11	김형준	26 25 35 89	186
12	김수아	25 26 35 92	190
13	원나재	26 26 35 90	190
14	노준역	25 25 35 95	194
15	기민석	26 26 35 98	200
16	문영진	26 26 36 89	193
17	민진우	25 25 33 88	188
18	여민이	26 26 34 89	193
19	정규환	25 26 35 88	193
20	박진주	25 26 35 88	194

Advanced WPF Programming

WPF Excel 파일 연동



Advanced WPF Programming

WPF Excel 파일 연동

	A	B	C	D	E	F	G	H
1		번호	이름	Sub1	Sub2	Sub3	Sub4	
2		1	강호민	25	25	36	90	
3		2	재석환	25	25	35	89	
4		3	고신진	25	25	35	90	
5		4	김동준	25	25	36	98	
6		5	김성만	26	26	36	90	
7		6	최민혁	26	26	36	95	
8		7	김병우	26	25	35	89	
9		8	김아영	26	26	35	87	
10		9	김지룡	25	26	35	89	
11		10	장성곤	25	25	36	88	
12		11	김형준	26	25	35	89	
13		12	김수아	25	26	35	92	
14		13	원나재	26	26	35	90	
15		14	노준역	25	25	35	95	
16		15	기민석	26	26	35	98	
17		16	문영진	26	26	36	89	
18		17	민진우	25	25	33	88	
19		18	여민이	26	26	34	89	
20		19	정규환	25	26	35	88	
21		20	박진주	25	26	35	88	
22		21	서동준	24	24	34	89	
23		22	김정인	25	25	33	88	
24		23	이석철	25	25	36	90	
25		24	안동이	25	25	33	89	
26		25	이진용	25	26	36	91	
27		26	김지용	26	26	34	90	
28		27	이영도	25	25	35	90	
29		28	현민석	26	26	36	90	
30		29	정훈민	26	25	36	89	
31		30	김진창	25	26	36	94	
32								
33								

	A	B	C	D	E	F	G
1	번호	이름	Sub1	Sub2	Sub3	Sub4	Sum
2	1	강호민	25	25	36	90	177
3	2	재석환	25	25	35	89	176
4	3	고신진	25	25	35	90	178
5	4	김동준	25	25	36	98	188
6	5	김성만	26	26	36	90	183
7	6	최민혁	26	26	36	95	189
8	7	김병우	26	25	35	89	182
9	8	김아영	26	26	35	87	182
10	9	김지룡	25	26	35	89	184
11	10	장성곤	25	25	36	88	184
12	11	김형준	26	25	35	89	186
13	12	김수아	25	26	35	92	190
14	13	원나재	26	26	35	90	190
15	14	노준역	25	25	35	95	194
16	15	기민석	26	26	35	98	200
17	16	문영진	26	26	36	89	193
18	17	민진우	25	25	33	88	188
19	18	여민이	26	26	34	89	193
20	19	정규환	25	26	35	88	193
21	20	박진주	25	26	35	88	194
22	21	서동준	24	24	34	89	192
23	22	김정인	25	25	33	88	193
24	23	이석철	25	25	36	90	199
25	24	안동이	25	25	33	89	196
26	25	이진용	25	26	36	91	203
27	26	김지용	26	26	34	90	202
28	27	이영도	25	25	35	90	202
29	28	현민석	26	26	36	90	206
30	29	정훈민	26	25	36	89	205
31	30	김진창	25	26	36	94	211

Advanced WPF Programming

WPF Excel 파일 연동

- UI 설계
- 참고 Code 이용하여 Excel 파일 Load 부터 Save까지 적용

Advanced WPF Programming

WPF Excel 파일 연동

```
private void LoadExcel_Click(object sender, RoutedEventArgs e)
{
    // 파일 열기 대화 상자를 사용하여 Excel 파일을 선택하도록 함
    OpenFileDialog openFileDialog = new OpenFileDialog();
    openFileDialog.Filter = "Excel Files|*.xlsx;*.xls";
    if (openFileDialog.ShowDialog() == true)
    {
        // 선택된 파일을 읽어 DataGrid에 표시
        string filePath = openFileDialog.FileName;
        DataTable dt = ReadExcelFile(filePath);
        dataGrid.ItemsSource = dt.DefaultView;
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동

```
// Excel 파일을 읽어서 DataTable로 변환하는 함수
private DataTable ReadExcelFile(string filePath)
{
    DataTable dt = new DataTable();

    // ClosedXML을 사용하여 Excel 파일을 읽음
    using (var workbook = new XLWorkbook(filePath))
    {
        var worksheet = workbook.Worksheet(1); // 첫 번째 워크시트
        bool firstRow = true;

        foreach (var row in worksheet.RowsUsed())
        {
            // 첫 번째 행은 DataTable의 컬럼을 설정
            if (firstRow)
            {
                ...
            }
        }
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동

계속...

```
        foreach (var cell in row.Cells())
        {
            dt.Columns.Add(cell.Value.ToString()); // 컬럼 이름 추가
        }
        firstRow = false;
    }
    else
    {
        // 그 외의 행은 데이터로 추가
        DataRow newRow = dt.NewRow();
        int columnIndex = 0;
        foreach (var cell in row.Cells())
        {
            newRow[columnIndex] = cell.Value.ToString();
            columnIndex++;
        }
        dt.Rows.Add(newRow);
    }
}
return dt;
}
```


Advanced WPF Programming

WPF Excel 파일 연동

- 파일 저장 기능 구현부

```
// DataGrid의 데이터를 새로운 Excel 파일로 저장하는 함수
private void SaveExcel_Click(object sender, RoutedEventArgs e)
{
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "Excel Files|*.xlsx";
    saveFileDialog.Title = "Save an Excel File";
    if (saveFileDialog.ShowDialog() == true)
    {
        string filePath = saveFileDialog.FileName;
        SaveDataGridToExcel(filePath);
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동

- 파일 저장 기능 구현부

```
// DataGrid의 데이터를 Excel 파일로 저장하는 함수
private void SaveDataGridToExcel(string filePath)
{
    using (var workbook = new XLWorkbook())
    {
        var worksheet = workbook.Worksheets.Add("Sheet1");

        // DataTable에서 컬럼 추가
        for (int col = 0; col < DataTable.Columns.Count; col++)
        {
            worksheet.Cell(1, col + 1).Value = DataTable.Columns[col].ColumnName;
        }
    }
}
```

계속...

Advanced WPF Programming

WPF Excel 파일 연동

- 파일 저장 기능 구현부

```
// DataTable에서 데이터 추가
for (int row = 0; row < DataTable.Rows.Count; row++)
{
    for (int col = 0; col < DataTable.Columns.Count; col++)
    {
        var cellValue = DataTable.Rows[row][col];

        // cellValue가 null이 아닌 경우 적절한 형식으로 캐스팅
        if (cellValue != DBNull.Value)
        {
            // 데이터 유형에 따라 명시적으로 변환
            worksheet.Cell(row + 2, col + 1).Value = cellValue.ToString(); // 문자열로 변환하여 할당
        }
        else
        {
            worksheet.Cell(row + 2, col + 1).Value = ""; // null 값일 경우 빈 문자열로 할당
        }
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동

- 파일 저장 기능 구현부

```
// DataGrid의 데이터를 Excel 파일로 저장하는 함수
private void SaveDataGridToExcel(string filePath)
{
    using (var workbook = new XLWorkbook())
    {
        var worksheet = workbook.Worksheets.Add("Sheet1");

        // DataTable에서 컬럼 추가
        for (int col = 0; col < DataTable.Columns.Count; col++)
        {
            worksheet.Cell(1, col + 1).Value = DataTable.Columns[col].ColumnName;
        }
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동

- 파일 저장 기능 구현부

```
// Excel 파일로 저장
    workbook.SaveAs(filePath);
}

    MessageBox.Show("Excel 파일이 성공적으로 저장되었습니다!", "저장 완료",
        MessageBoxButton.OK, MessageBoxImage.Information);
}
```

Advanced WPF Programming

WPF Excel 파일 연동

- Sum 컬럼 추가
 - 기존 컬럼들 끝에 Sum 컬럼 새로 추가
 - 각 행의 값을 더해서 sum 컬럼에 계산 값 저장

```
// 더하기 버튼 클릭 시 호출되는 함수
private void SumButton_Click(object sender, RoutedEventArgs e)
{
    // 이미 "Sum" 컬럼이 존재하는 경우 제거
    if (DataTable.Columns.Contains("Sum"))
    {
        DataTable.Columns.Remove("Sum");
    }

    // DataTable에 "Sum" 컬럼 추가
    DataTable.Columns.Add("Sum", typeof(double));
}
```

Advanced WPF Programming

WPF Excel 파일 연동

- Sum 컬럼 추가

```
// 각 행의 숫자 값 합계를 계산하여 "Sum" 컬럼에 추가
foreach (DataRow row in DataTable.Rows)
{
    double sum = 0;

    // 각 컬럼의 값을 더함 (마지막 컬럼 제외)
    for (int i = 0; i < DataTable.Columns.Count - 1; i++)
    {
        if (double.TryParse(row[i].ToString(), out double value))
        {
            sum += value;
        }
    }

    // 계산된 합계를 "Sum" 컬럼에 넣음
    row["Sum"] = sum;
}

// DataGridView의 ItemsSource를 다시 설정하여 업데이트 반영
dataGridView.ItemsSource = null; // 먼저 ItemsSource를 null로 설정하여 데이터 새로고침
dataGridView.ItemsSource = DataTable.DefaultView; // 다시 DataTable로 설정
}
```

Advanced WPF Programming

WPF Excel 파일 연동

Advanced WPF Programming

WPF Excel 파일 연동

- Syncfusion XlsIO 사용하기

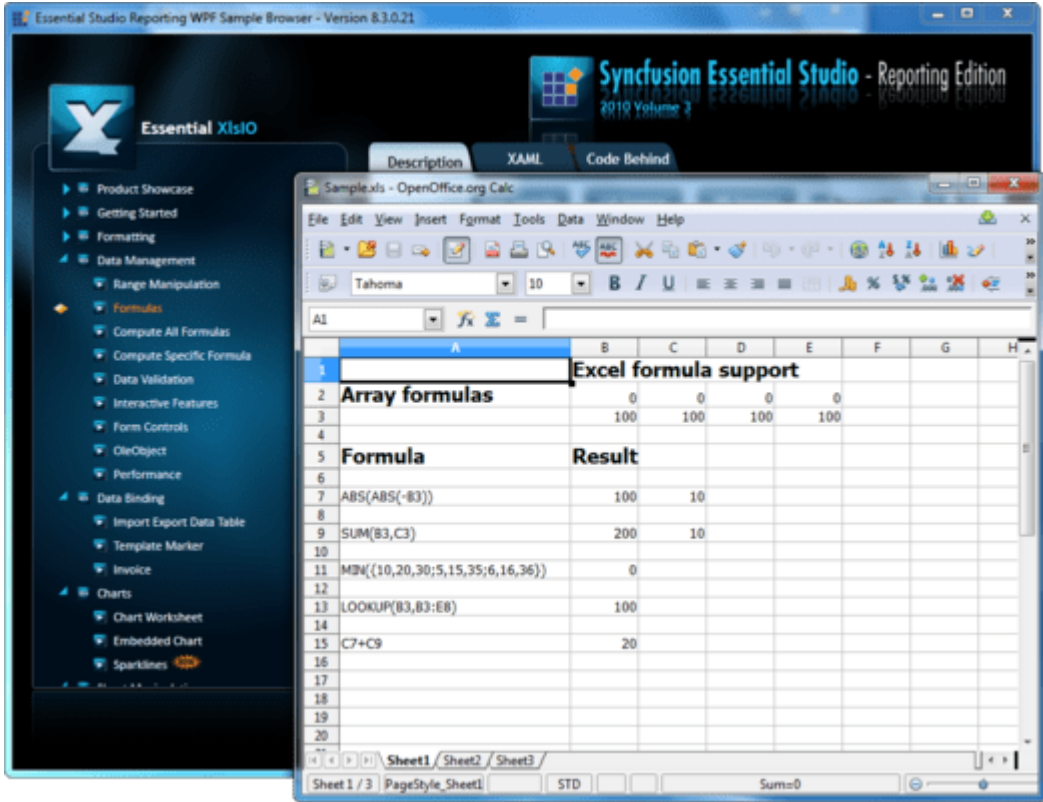
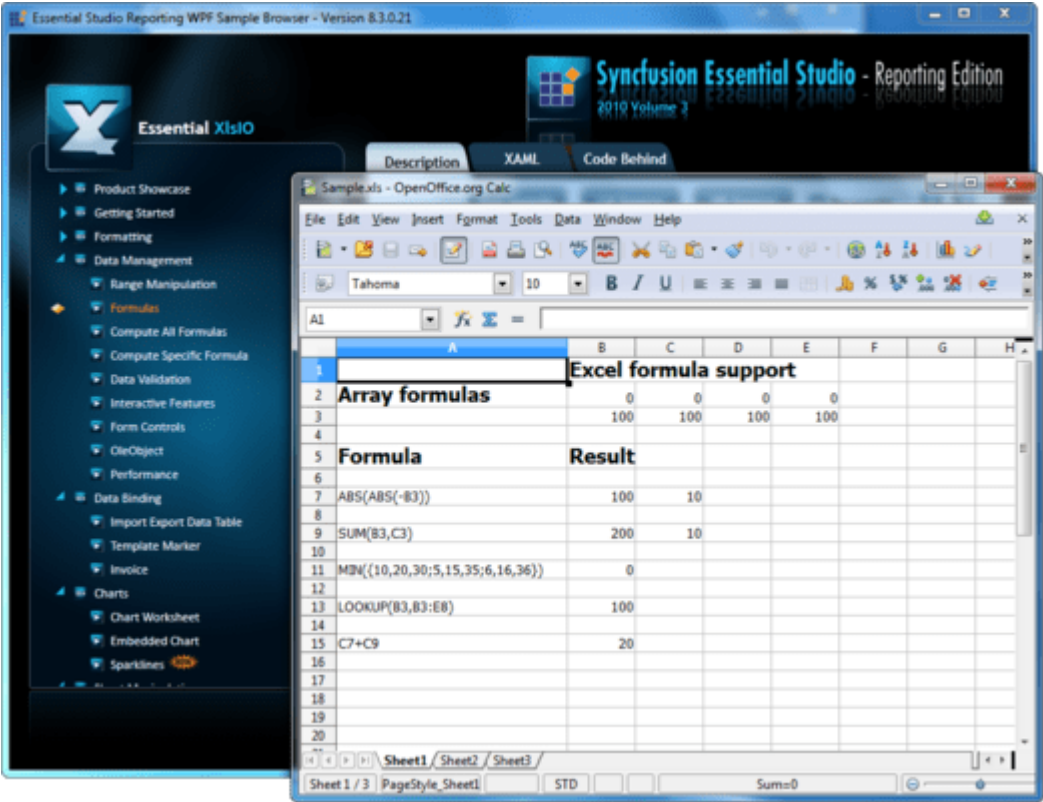
The screenshot displays the Visual Studio IDE with the following components:

- NuGet - 솔루션* (Solution):** The '찾아보기' (Browse) tab is active, showing a list of Syncfusion.XlsIO packages. 'Syncfusion.XlsIO.Wpf' is selected, with version 27.1.51. The package description states: 'This package provides the functionality to utilize the features of Syncfusion WPF Excel library and more.'
- 솔루션 패키지 관리 (Solution Package Management):** The 'Syncfusion.XlsIO.Wpf' package is listed under '버전 - 0' (Version - 0). The '설치됨' (Installed) checkbox is checked. The '설치됨' (Installed) button is disabled, and the '제거' (Remove) button is visible.
- 솔루션 탐색기 (Solution Explorer):** The 'WpfExcelCreate' project is expanded, showing files: 'Properties', '참조' (References), 'App.config', 'App.xaml', and 'MainWindow.xaml'.
- 변경 내용 미리 보기 (Preview Changes):** A dialog box is open, showing the changes for the 'WpfExcelCreate' project. The '설치 중' (Installing) section lists the following packages and versions: 'Syncfusion.Compression.Base.27.1.51', 'Syncfusion.Licensing.27.1.51', and 'Syncfusion.XlsIO.Wpf.27.1.51'.
- 파일 탐색기 (File Explorer):** The 'WpfExcelCreate' project is selected, showing the following files: 'Properties', '참조' (References), 'App.config', 'App.xaml', 'MainWindow.xaml', 'MainWindow.xaml.cs', and 'packages.config'.

- Syncfusion Essential XlsIO for WPF
 - 비즈니스 어플리케이션을 위한 세련된 Excel의 레포팅.
 - Read / Write : Microsoft Excel파일을 읽고 쓰기를 합니다.
 - 버전 : Office 97에 포함되어 있는 Excel이후의 모든 버전에 대응합니다.
 - 오브젝 모델 : Microsoft Office Excel 오토메이션 모델에 필적하는 완벽한 오브젝 모델을 제공합니다.
 - 100% Managed Code : C#를 이용하여 스크리치를 구축하고, COM상호 운용 성을 갖지 않습니다.
 - Excel 비의존 : Excel이 인스톨 되어있지 않은 환경에서도 이용이 가능합니다.

- Syncfusion Essential XlsIO for WPF
 - 스프레드 시트 작성 특징
 - 멀티 워크시트 : 복수의 워크시트로 구성되는 워크북을 작성합니다.
 - 템플레이트 : 템플레이트를 통하여 워크북을 작성합니다. (기존의 워크북) .
 - 텍스트 포맷 : 셀 내의 컬러, 타입, 스타일, 폰트를 설정하고 변경합니다.
 - 수치 포맷 : 셀 내에서 소수 부분, 퍼센트, 통화, 일자를 표시합니다.
 - 셀의 음영 : 셀의 색 채우기나 패턴 적용으로 음영을 만듭니다.
 - 셀의 머지와 해소를 실행합니다.
 - 셀의 그룹화와 해소 : Row과 Column에 의하여 셀을 그룹화하고 그것을 해소합니다.
 - 스타일 : 스타일을 이용하여 복잡한 포맷을 적용합니다.
 - 렌지(range) 이름 설정 : 이름이 설정된 렌지를 작성하여 조작합니다.
 - 수식의 서포트 : 정의 완료된 Excel함수와 커스텀 수식을 사용합니다.
 - 스프레드시트의 수정 : 기존의 스프레드시트를 읽어 들이고 수정합니다.

WPF Excel 파일 연동



Advanced WPF Programming

WPF Excel 파일 연동

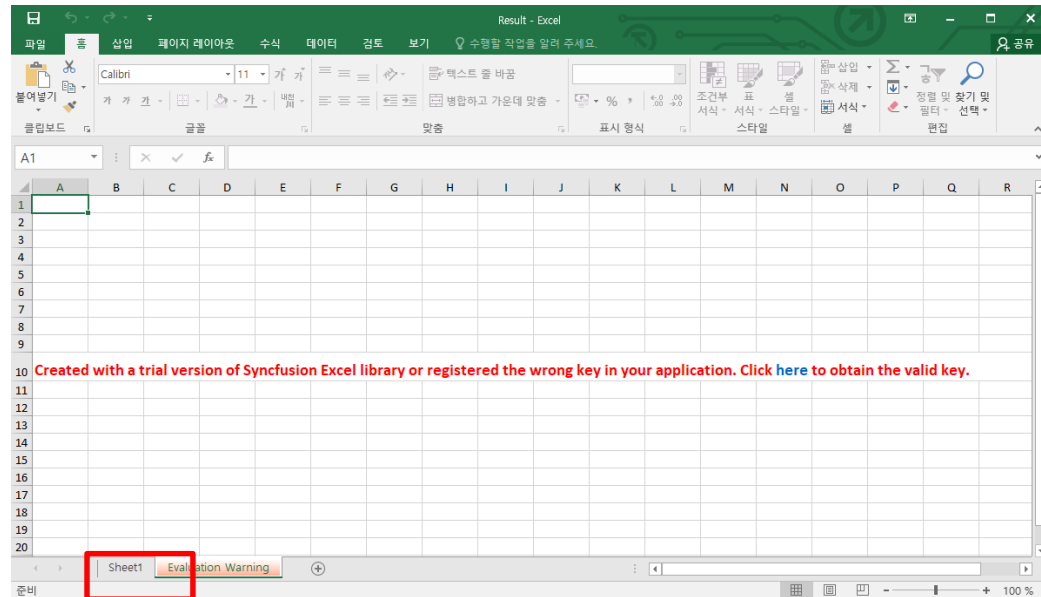
```
private void btnCreate_Click(object sender, RoutedEventArgs e)
{
    using (ExcelEngine engine = new ExcelEngine())
    {
        IApplication application = engine.Excel;
        application.DefaultVersion = ExcelVersion.Excel2016;

        IWorkbook workbook = application.Workbooks.Create(1);
        IWorksheet worksheet = workbook.Worksheets[0];

        workbook.SaveAs("Result.xlsx");
        this.Close();
        System.Diagnostics.Process.Start("Result.xlsx");
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동



Advanced WPF Programming

WPF Excel 파일 연동

- 이미지 추가

```
private void btnCreate_Click(object sender, RoutedEventArgs e)
{
    using (ExcelEngine engine = new ExcelEngine())
    {
        IApplication application = engine.Excel;
        application.DefaultVersion = ExcelVersion.Excel2016;

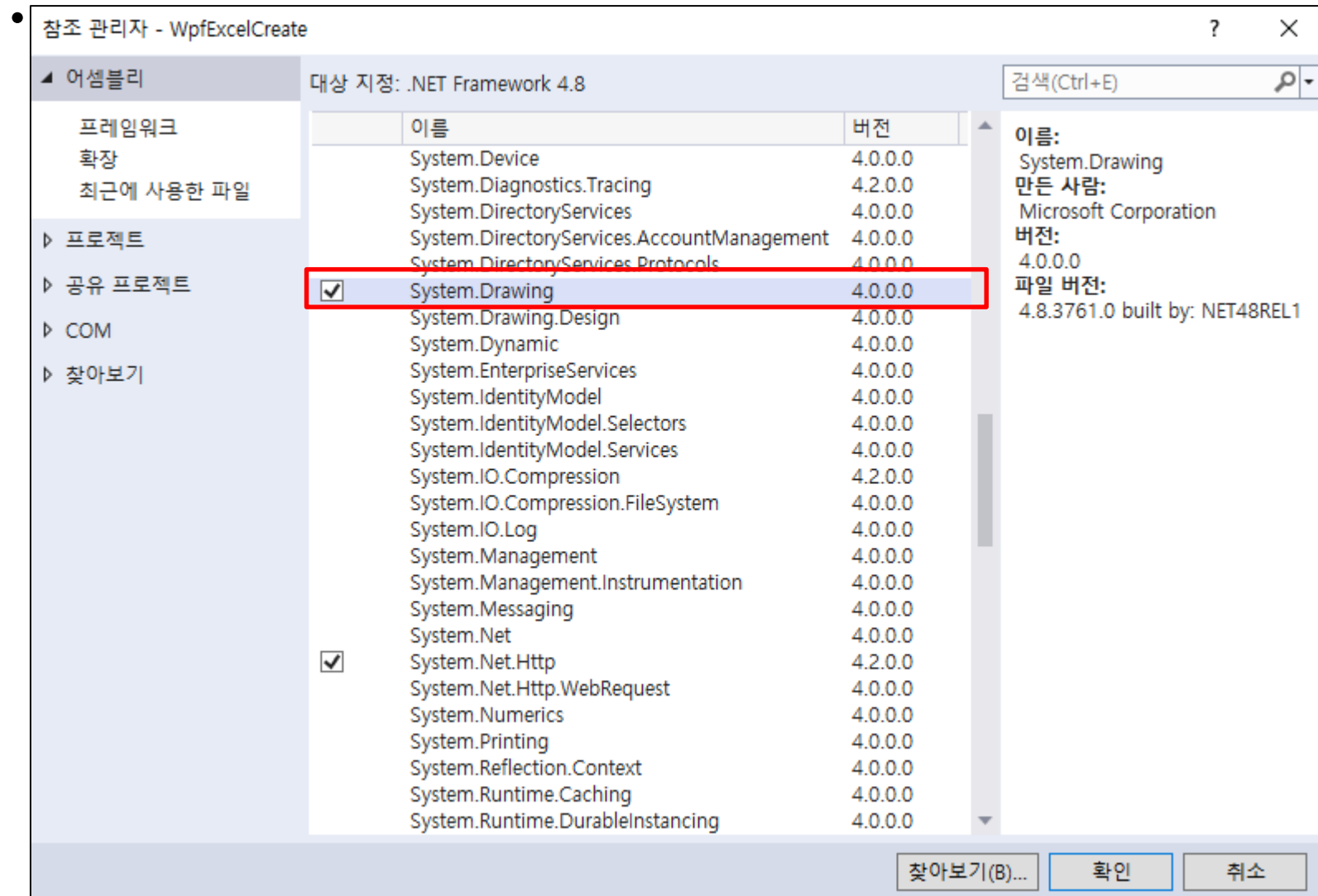
        IWorkbook workbook = application.Workbooks.Create(1);
        IWorksheet worksheet = workbook.Worksheets[0];

        IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, @"D:\Wnetpage.PNG");

        workbook.SaveAs("Result.xlsx");
        this.Close();
        System.Diagnostics.Process.Start("Result.xlsx");
    }
}
```

Advanced WPF Programming

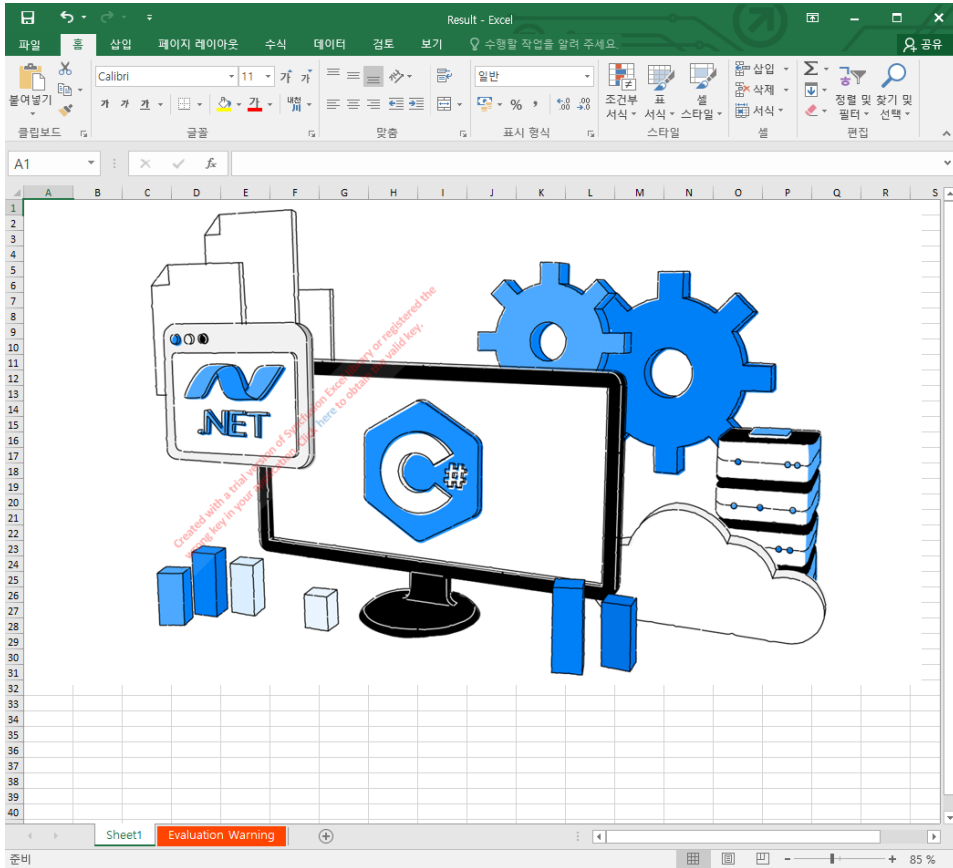
WPF Excel 파일 연동



Advanced WPF Programming

WPF Excel 파일 연동

- Excel 파일에 이미지 표현



Advanced WPF Programming

WPF Excel 파일 연동

- 이미지 추가

```
private void btnCreate_Click(object sender, RoutedEventArgs e)
{
    using (ExcelEngine engine = new ExcelEngine())
    {
        IApplication application = engine.Excel;
        application.DefaultVersion = ExcelVersion.Excel2016;

        IWorkbook workbook = application.Workbooks.Create(1);
        IWorksheet worksheet = workbook.Worksheets[0];

        worksheet.IsGridLinesVisible = false;
        IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, @"D:\Wnetpage.PNG");

        workbook.SaveAs("Result.xlsx");
        this.Close();
        System.Diagnostics.Process.Start("Result.xlsx");
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동

- 이미지 추가

```
private void btnCreate_Click(object sender, RoutedEventArgs e)
{
    using (ExcelEngine engine = new ExcelEngine())
    {
        IApplication application = engine.Excel;
        application.DefaultVersion = ExcelVersion.Excel2016;

        IWorkbook workbook = application.Workbooks.Create(1);
        IWorksheet worksheet = workbook.Worksheets[0];

        worksheet.IsGridLinesVisible = false;

        worksheet.Range["A3"].Text = "Song";
        worksheet.Range["A4"].Text = "Park";
        worksheet.Range["A5"].Text = "Kim";
        IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, @"D:\Wnetpage.PNG");

        workbook.SaveAs("Result.xlsx");
        this.Close();
        System.Diagnostics.Process.Start("Result.xlsx");
    }
}
```

	A	B	C	D
1				
2				
3	Song			
4	Park			
5	Kim			

Advanced WPF Programming

WPF Excel 파일 연동

- using Color = System.Drawing.Color;

```
worksheet.Range["D1:E1"].Merge();

worksheet.Range["D1"].Text = "test";
worksheet.Range["D1"].CellStyle.Font.Bold = true;
worksheet.Range["D1"].CellStyle.Font.RGBColor = Color.FromArgb(100, 120, 44);

worksheet.Range["D5"].Text = "wpf";
worksheet.Range["E5"].Text = "Date";

worksheet.Range["D6"].Number = 1024;
worksheet.Range["E6"].Value = "2024-10-10";

worksheet.Range["D7"].Text = "id";
worksheet.Range["E7"].Text = "terms";

worksheet.Range["D8"].Number = 1111;
worksheet.Range["E8"].Value = "202";

workbook.SaveAs("Result.xlsx");
this.Close();
System.Diagnostics.Process.Start("Result.xlsx");
```

	A	B	C	D	E	F
2						
3	Song					
4	Park					
5	Kim			wpf	Date	
6				1024	2024-10-10	
7				id	terms	
8				1111	202	
9						

Advanced WPF Programming

WPF Excel 파일 연동

```
private void btnCreate_Click(object sender, RoutedEventArgs e)
{
    using (ExcelEngine engine = new ExcelEngine())
    {
        IApplication application = engine.Excel;
        application.DefaultVersion = ExcelVersion.Excel2016;

        IWorkbook workbook = application.Workbooks.Create(1);
        IWorksheet worksheet = workbook.Worksheets[0];

        worksheet.IsGridLinesVisible = false;

        worksheet.Range["A3"].Text = "Song";
        worksheet.Range["A4"].Text = "Park";
        worksheet.Range["A5"].Text = "Kim";

        IPictureShape shape = worksheet.Pictures.AddPicture(1, 1, @"D:\Wnetpage.PNG");

        worksheet.Range["D1:E1"].Merge();

        worksheet.Range["D1"].Text = "test";
        worksheet.Range["D1"].CellStyle.Font.Bold = true;
        worksheet.Range["D1"].CellStyle.Font.RGBColor = Color.FromArgb(100, 120, 44);

        worksheet.Range["D1"].CellStyle.HorizontalAlign = ExcelHAlign.HAlignRight;
        worksheet.Range["D1"].CellStyle.VerticalAlign = ExcelVAlign.VAlignTop;
    }
}
```

Advanced WPF Programming

WPF Excel 파일 연동

... 계속

```
worksheet.Range["D5"].Text = "wpf";  
worksheet.Range["E5"].Text = "Date";
```

```
worksheet.Range["D6"].Number = 1024;  
worksheet.Range["E6"].Value = "2024-10-10";
```

```
worksheet.Range["D7"].Text = "id";  
worksheet.Range["E7"].Text = "terms";
```

```
worksheet.Range["D8"].Number = 1111;  
worksheet.Range["E8"].Value = "202";
```

```
worksheet.Range["D5:E8"].CellStyle.HorizontalAlign = ExcelHAlign.HAlignCenter;  
worksheet.Range["D5:E5"].CellStyle.VerticalAlign = ExcelVAlign.VAlignCenter;  
worksheet.Range["D7:E7"].CellStyle.VerticalAlign = ExcelVAlign.VAlignCenter;  
worksheet.Range["D6:E6"].CellStyle.VerticalAlign = ExcelVAlign.VAlignTop;
```

Advanced WPF Programming

WPF Excel 파일 연동

... 계속

```
worksheet.Range["A7"].Text = "BILL To";
worksheet.Range["A7"].CellStyle.Color = Color.FromArgb(43, 120, 230);
worksheet.Range["A7"].CellStyle.Font.Bold = true;
worksheet.Range["A7"].CellStyle.Font.Color = ExcelKnownColors.White;

worksheet.Range["A7"].CellStyle.HorizontalAlign = ExcelHAlign.HAlignLeft;
worksheet.Range["A7"].CellStyle.VerticalAlign = ExcelVAlign.VAlignCenter;

worksheet.Range["A8"].Text = "TESTTEST";
worksheet.Range["A9"].Text = "TESTTEST";
worksheet.Range["A10"].Text = "TESTTEST";
worksheet.Range["A11"].Text = "TESTTEST";
worksheet.Range["A12"].Text = "TESTTEST";

IHyperLink hyperlink = worksheet.HyperLinks.Add(worksheet.Range["A13"]);
hyperlink.Type = ExcelHyperLinkType.Url;
hyperlink.Address = "test@gmail.com";
hyperlink.ScreenTip = "Send Mail";

workbook.SaveAs("Result.xlsx");
this.Close();
System.Diagnostics.Process.Start("Result.xlsx");
}
```

Advanced WPF Programming

WPF Excel 파일 연동

- Excel 파일 생성 결과

	A	B	C	D	E	F	G	H	I	J	K	L	M
1					test								
2													
3	Song												
4	Park												
5	Kim												
6		wpf			Date								
7	BILL To	1024			2024-10-10								
8	TESTTEST	id			terms								
9	TESTTEST	1111			202								
10	TESTTEST												
11	TESTTEST												
12	TESTTEST												
13	test@gmail.com												
14													
15													
16													
17													
18													
19													



Advanced WPF Programming

WPF JSON 파일 연동

- JSON 파일을 읽고 쓸 수 있는 라이브러리 종류

라이브러리 이름	설명	설치 방법	유/무료 사용
Newtonsoft.Json (Json.NET)	가장 많이 사용되는 JSON 처리 라이브러리	Install-Package Newtonsoft.Json	무료
System.Text.Json	.NET Core 3.0 이상에서 제공되는 내장 JSON 라이브러리	SDK에 포함되어 있음	무료
Jil	매우 빠른 JSON 직렬화 및 역직렬화 제공	Install-Package Jil	무료
Utf8Json	고속 JSON 직렬화 및 역직렬화 라이브러리	Install-Package Utf8Json	무료
Json.NET for .NET Standard	Newtonsoft.Json의 .NET Standard 버전	라이선스 확인 필요	유료
Xceed Toolkit Plus	JSON 관련 기능이 포함된 WPF 컨트롤 및 라이브러리	Xceed 공식 웹사이트	유료
Telerik UI for WPF	JSON 데이터와 통합 기능이 포함된 UI 컴포넌트	Telerik 공식 웹사이트	유료

Advanced WPF Programming

WPF JSON 파일 연동

- Newtonsoft.Json 사용하기

The screenshot shows the Visual Studio IDE with the NuGet Package Manager window open. The package 'Newtonsoft.Json' is selected, and its details are displayed on the right. A dialog box titled '변경 내용 미리 보기' (Preview Changes) is open, showing the package 'WpfJSONFile' with the installed version 'Newtonsoft.Json.13.0.3'. The dialog box also includes a '복사(P)' (Copy) button and a '다시 표시하지 않음(D)' (Don't show again) checkbox.

NuGet 패키지 관리자: WpfJSONFile

찾아보기 | 설치됨 | 업데이트

Newtonsoft.Json x [refresh] [checkbox checked] 시험판 포함 패키지 소스: nuget.org

Newtonsoft.Json [verified] 작성자: James 13.0.3
Json.NET is a popular high-performance JSON framework for .NET

Newtonsoft.Json.Bson [verified] 작 1.0.3-beta1
[시험판] Json.NET Bson adds support for reading and writing BSON

Newtonsoft.Json.Schema [verified] 4.0.2-beta1
[시험판] Json.NET Schema is a complete and easy-to-use JSON Schema fr...

Microsoft.AspNetCore 9.0.0-rc.2.24474.3
ASP.NET Core MVC features

각 패키지는 해당 소유자에 의해 사용이 허가되었습니다. NuGet은 타사 패키지에 대해 책임을 지지 않으며 라이선스를 부여하지도 않습니다.

☐ 다시 표시하지 않음(D)

버전: 안정적인 최신 버전 13.0.3 [설치]

패키지 원본 매핑이 꺼져 있습니다. 구성

옵션

설명
Json.NET is a popular high-performance JSON framework for .NET

버전: 13.0.3
작성자: James Newton-King
라이선스: MIT
추가 정보: 추가 정보 보기
다운로드: 5,312,054,208
게시 날짜: 2023년 3월 8일 수요일 (2023-03-08)
프로젝트 URL: <https://www.newtonsoft.com/json>

변경 내용 미리 보기

Visual Studio에서 이 솔루션을 변경하려고 합니다. [복사(P)]

WpfJSONFile
설치 중:
Newtonsoft.Json.13.0.3

☐ 다시 표시하지 않음(D) [적용(A)] [취소(C)]

Advanced WPF Programming

WPF JSON 파일 연동

- 이름, 나이를 리스트에 작성

The screenshot shows the 'MainWindow' application. It features a table with two columns: 'Name' and 'Age'. Below the table, there are two empty text input fields. At the bottom, there are three buttons: 'Add to List', 'Save to JSON', and 'Load from JSON'.

Name	Age
------	-----

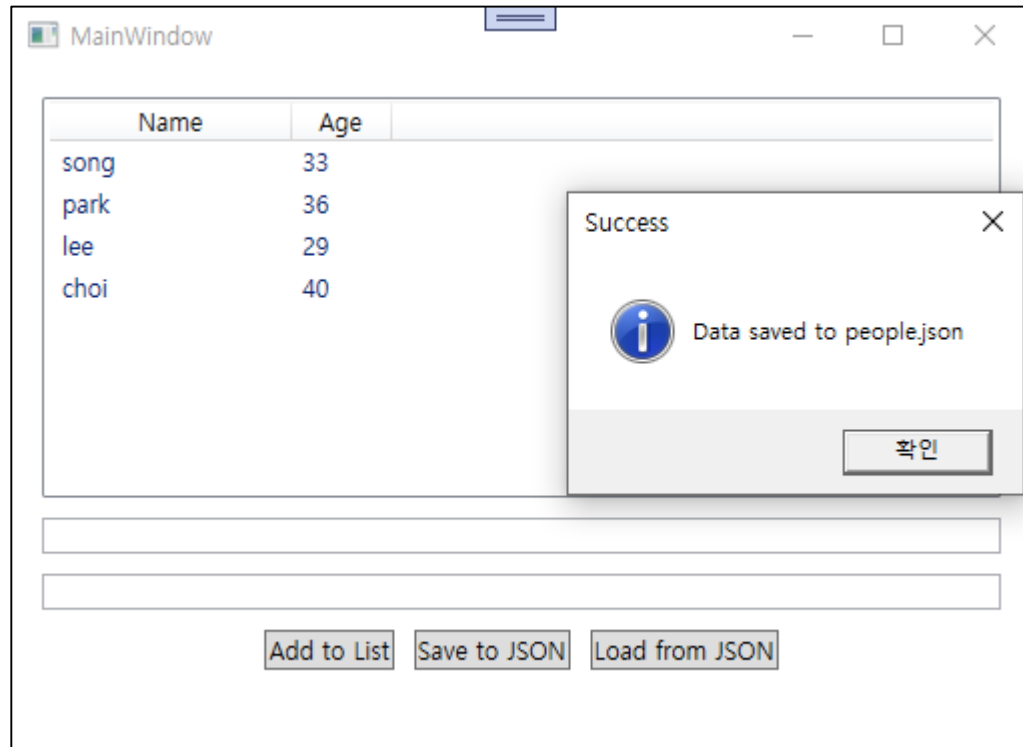
The screenshot shows the 'MainWindow' application with the table populated with four entries. Below the table, there are two empty text input fields. At the bottom, there are three buttons: 'Add to List', 'Save to JSON', and 'Load from JSON'.

Name	Age
song	33
park	36
lee	29
choi	40

Advanced WPF Programming

WPF JSON 파일 연동

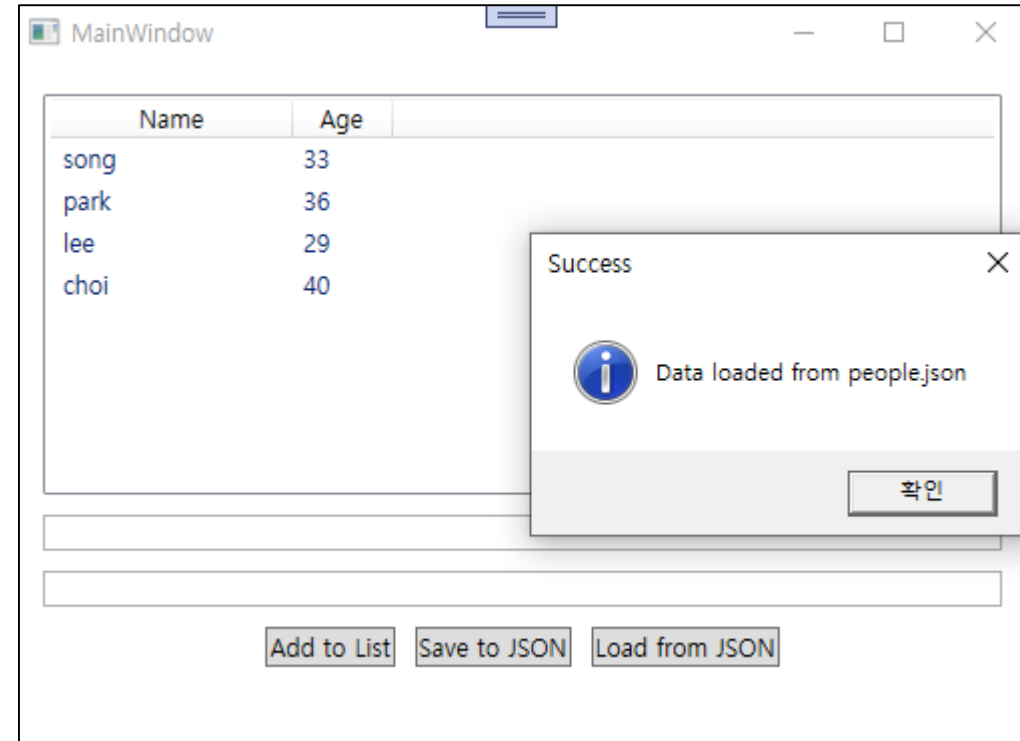
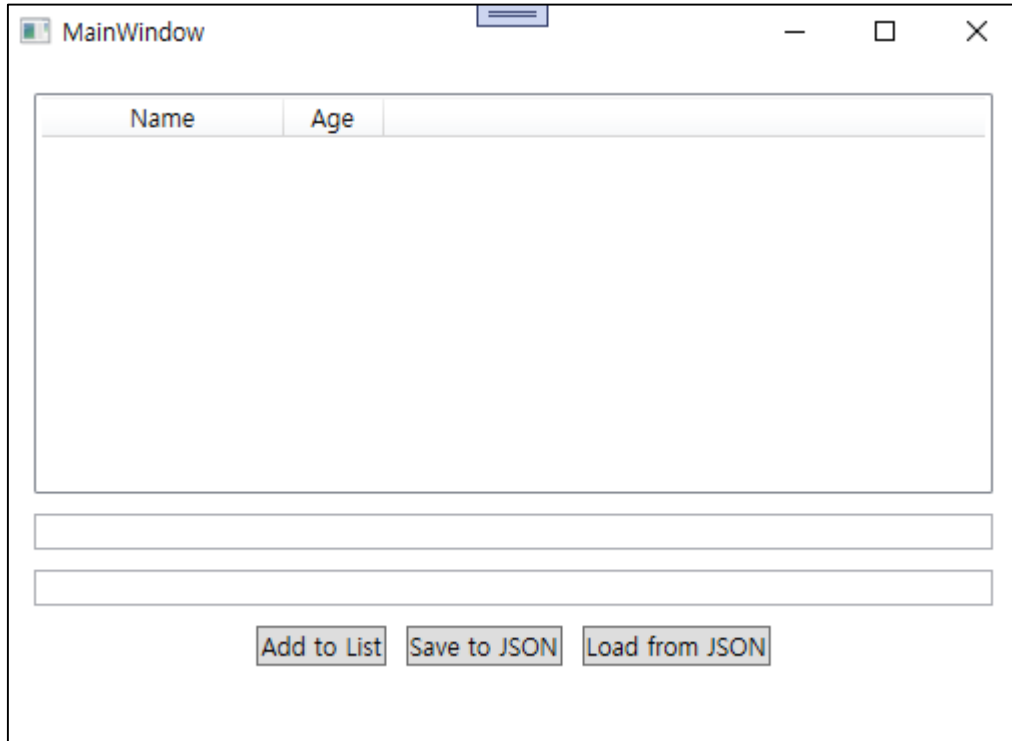
- ListView 내용을 JSON 파일로 저장하기



Advanced WPF Programming

WPF JSON 파일 연동

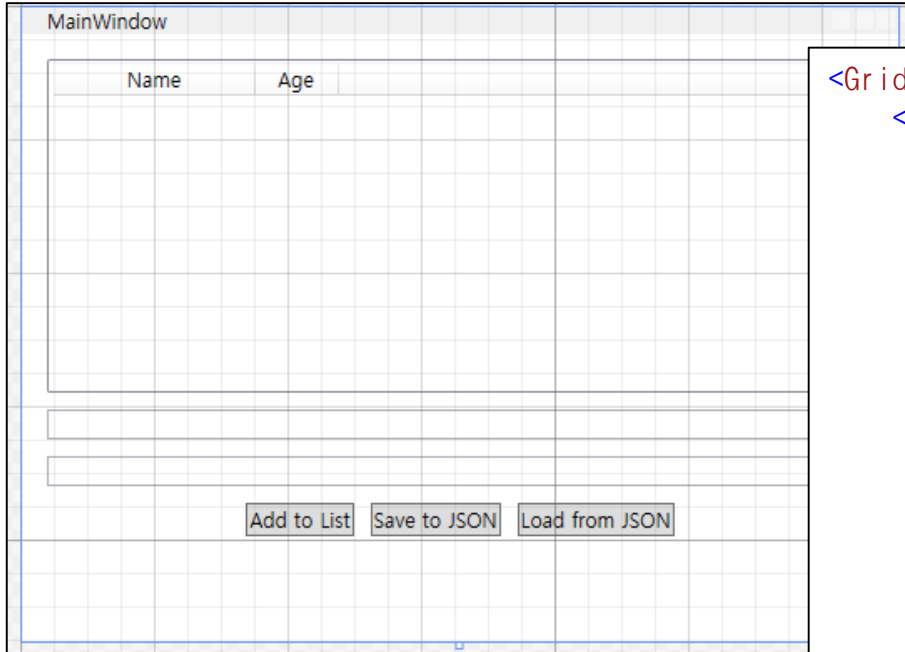
- 프로그램 재실행 후 빈 ListView에 JSON파일 읽어오기 기능



Advanced WPF Programming

WPF JSON 파일 연동

- UI 디자인



```
<Grid Margin="10">
    <StackPanel>
        <ListView x:Name="MyListView" Margin="5" Height="200">
            <ListView.View>
                <GridView>
                    <GridViewColumn Header="Name" Width="120" DisplayMemberBinding="{Binding Name}" />
                    <GridViewColumn Header="Age" Width="50" DisplayMemberBinding="{Binding Age}" />
                </GridView>
            </ListView.View>
        </ListView>
        <TextBox x:Name="NameTextBox" Margin="5" />
        <TextBox x:Name="AgeTextBox" Margin="5" />
        <StackPanel Orientation="Horizontal" HorizontalAlignment="Center">
            <Button Content="Add to List" Click="AddToList_Click" Margin="5" />
            <Button Content="Save to JSON" Click="SaveToJson_Click" Margin="5" />
            <Button Content="Load from JSON" Click="LoadFromJson_Click" Margin="5" />
        </StackPanel>
    </StackPanel>
</Grid>
```

Advanced WPF Programming

WPF JSON 파일 연동

```
public class Person
{
    public string Name { get; set; }
    public int Age { get; set; }
}
```

```
public List<Person> People { get; set; } = new List<Person>();
```

Advanced WPF Programming

WPF JSON 파일 연동

-

```
private void AddToList_Click(object sender, RoutedEventArgs e)
{
    if (!string.IsNullOrEmpty(NameTextBox.Text) &&
        int.TryParse(AgeTextBox.Text, out int age))
    {
        var person = new Person { Name = NameTextBox.Text, Age = age };
        People.Add(person);
        MyListView.Items.Add(person);
        NameTextBox.Clear();
        AgeTextBox.Clear();
    }
    else
    {
        MessageBox.Show( " Please enter valid Name and Age.", "Input Error",
                        MessageBoxButton.OK, MessageBoxImage.Warning);
    }
}
```


Advanced WPF Programming

WPF JSON 파일 연동

-

```
private void SaveToJson_Click(object sender, RoutedEventArgs e)
{
    var json = JsonConvert.SerializeObject(People, Formatting.Indented);

    File.WriteAllText("people.json", json);

    MessageBox.Show("Data saved to people.json", "Success",
                    MessageBoxButton.OK, MessageBoxImage.Information);
}
```

Advanced WPF Programming

WPF JSON 파일 연동

- ```
private void LoadFromJson_Click(object sender, RoutedEventArgs e)
{
 if (File.Exists("people.json"))
 {
 var json = File.ReadAllText("people.json");
 People = JsonConvert.DeserializeObject<List<Person>>(json);
 MyListView.Items.Clear();

 foreach (var person in People)
 {
 MyListView.Items.Add(person);
 }
 MessageBox.Show("Data loaded from people.json", "Success",
 MessageBoxButton.OK, MessageBoxImage.Information);
 }
 else
 {
 MessageBox.Show("people.json file not found.", "Error",
 MessageBoxButton.OK, MessageBoxImage.Error);
 }
}
```

[C# 기반]

Advanced WPF Programming

**WPF DATABASE 연동**

# Advanced WPF Programming

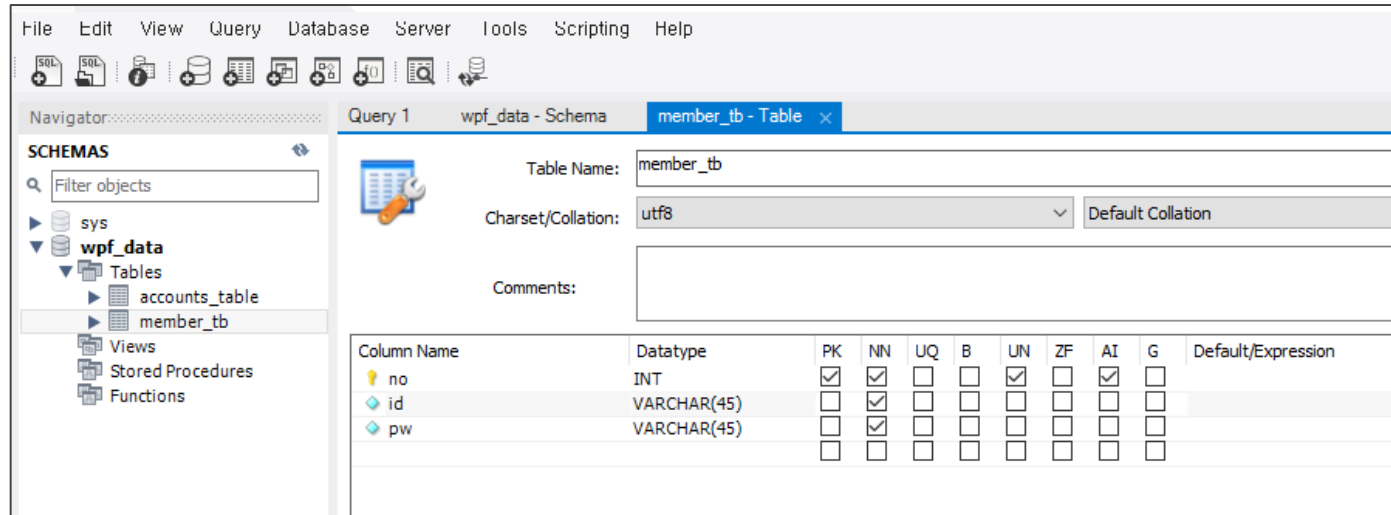
DataBase 연동(MySQL)

1. Mysql Server 및 Workbench 설치  
<https://dev.mysql.com/downloads/installer/>
2. Mysql.Data.dll 파일 설치 및 참조 추가  
<https://dev.mysql.com/downloads/connector/net/>
3. Mysql DataBase 테이블 생성
4. LoginDB WPF 프로젝트 생성
  - XAML을 이용한 UI 디자인
  - 필요한 로직 구현

# Advanced WPF Programming

DataBase 연동(MySQL)

- Table 생성

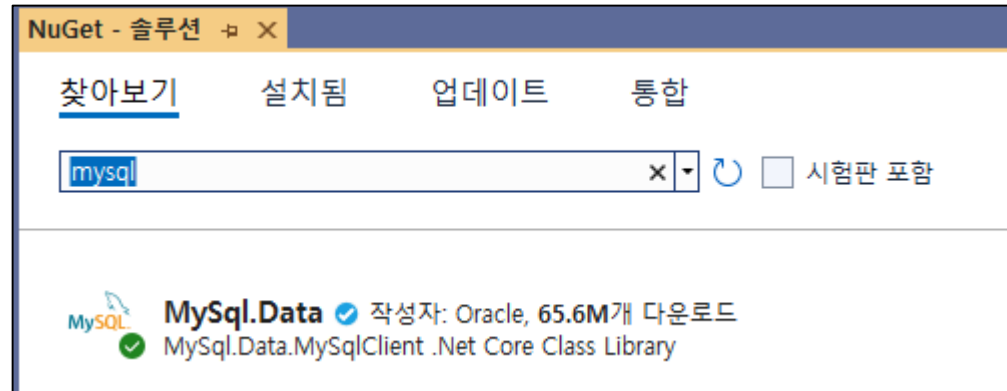
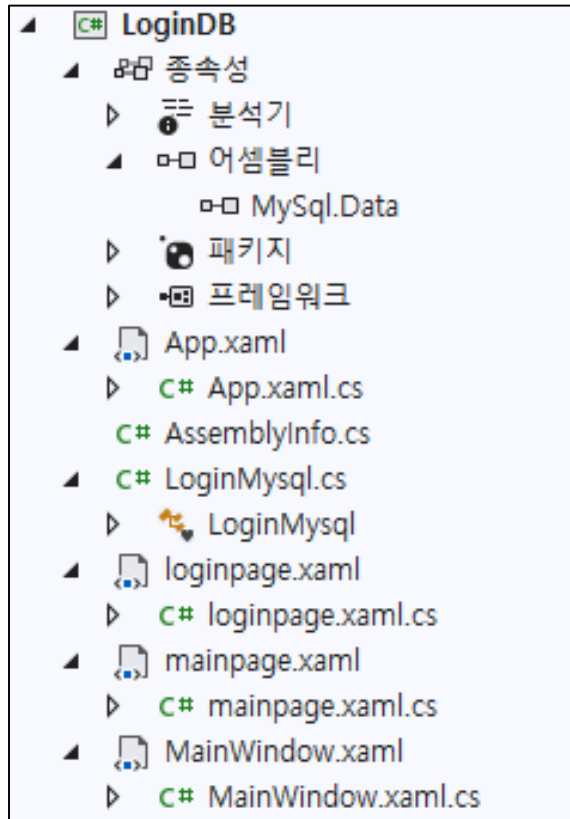


```
CREATE TABLE `wpf_data`.`signup_tb` (
 `no` INT UNSIGNED NOT NULL AUTO_INCREMENT,
 `id` VARCHAR(45) NOT NULL,
 `pw` VARCHAR(45) CHARACTER SET 'utf8' NOT NULL,
 PRIMARY KEY (`no`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

# Advanced WPF Programming

DataBase 연동(MySQL)

- 프로젝트 생성 및 MySql.Data.dll 라이브러리 추가
- 도구 -> NuGet 패키지 관리자 -> 솔루션용 NuGet 패키지 관리자 -> MySql.Data 설치



# Advanced WPF Programming

DataBase 연동(MySQL)

- MainWindow.xaml 파일에서 login page로 첫 화면 변경

```
<Window x:Class="LoginDB.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:LoginDB"
 mc:Ignorable="d"
 Title="MemberManagement" Height="450" Width="800">

 <Grid>
 <Frame Source="/loginpage.xaml" > </Frame>
 </Grid>
</Window>
```

# Advanced WPF Programming

DataBase 연동(MySQL)

- mainpage.xaml

```
<Page x:Name="page" x:Class="LoginDB.mainpage"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:local="clr-namespace:LoginDB"
 mc:Ignorable="d"
 d:DesignHeight="450" d:DesignWidth="800"
 FontSize="30"
 Title="mainpage">

 <Grid>
 <TextBlock x:Name="textBlock" HorizontalAlignment="Center" TextWrapping="Wrap" Text="Main Page로 이동하셨습니다."
 VerticalAlignment="Center" Background="{Binding Background, ElementName=page}" Height="50"/>

 </Grid>
</Page>
```



# Advanced WPF Programming

DataBase 연동(MySQL)

- loginpage.xaml

```
<Page x:Class="LoginDB.loginpage"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
 xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
 xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
 xmlns:local="clr-namespace:LoginDB"
 mc:Ignorable="d"
 d:DesignHeight="450" d:DesignWidth="800"
 Title="loginpage">
 <Page.Resources>
 <Style TargetType="TextBox" x:Key="tbStyle">
 <Setter Property="Height" Value="30"/>
 <Setter Property="VerticalContentAlignment" Value="Center"/>
 </Style>
 </Page.Resources>

 <Grid>
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="19*"/>
 <ColumnDefinition Width="141*"/>
 </Grid.ColumnDefinitions>
```

# Advanced WPF Programming

## DataBase 연동(MySQL)

- loginpage.xaml 계속

```
<GroupBox x:Name="gdInfo" Header="MySQL DataBase 연동" Margin="55,67,150,67"
 Height="300" Width="500" Grid.Column="1">
 <StackPanel Width="200" VerticalAlignment="Center">
 <StackPanel Margin="5">
 <Label Content="아이디"/>
 <TextBox x:Name="tbId" Style="{DynamicResource tbStyle}"/>
 </StackPanel>
 <StackPanel Margin="5">
 <Label Content="비밀번호"/>
 <TextBox x:Name="tbPw" Style="{DynamicResource tbStyle}"/>
 </StackPanel>
 <StackPanel Orientation="Horizontal" Margin="5" HorizontalAlignment="Center">
 <StackPanel.Resources>
 <Style TargetType="Button">
 <Setter Property="Background" Value="Transparent"/>
 <Setter Property="BorderBrush" Value="Transparent"/>
 <Setter Property="Height" Value="30"/>
 <Setter Property="Width" Value="70"/>
 </Style>
 </StackPanel.Resources>
 <StackPanel Orientation="Horizontal">
 <Button Content="회원가입" Margin="5" Click="btnSignUp_Click"/>
 <Button Content="로그인" Margin="5" Click="btnLogin_Click"/>
 </StackPanel>
 </StackPanel>
 </StackPanel>
</GroupBox>
</Grid>
</Page>
```

# Advanced WPF Programming

DataBase 연동(MySQL)

- loginpage.cs

```
public partial class loginpage : Page
{
 LoginMysql manager = new LoginMysql();
 public loginpage()
 {
 InitializeComponent();
 manager.Initialize();
 }

 public class SignUpEventArgs : EventArgs
 {
 public bool isSignUp;
 }

 private void btnSignUp_Click(object sender, RoutedEventArgs e)
 {
 SignUpEventArgs args = new SignUpEventArgs();

 string query = "INSERT INTO member_tb(id, pw)" + "VALUES('" + tbId.Text + "','" + tbPw.Text + "')";
 manager.MySqlQueryExecuter(query);

 if (App.DataSaveResult == true)
 {
 args.isSignUp = true;
 }

 if (args.isSignUp == true)
 {
 MessageBox.Show("회원이입에 성공하셨습니다!");
 tbId.Text = string.Empty;
 tbPw.Text = string.Empty;
 }
 else
 {
 MessageBox.Show("회원이입에 실패하셨습니다.");
 }
 }
}
```

# Advanced WPF Programming

DataBase 연동(MySQL)

- loginpage.cs 계속

```
public class LoginEventArgs : EventArgs
{
 public bool isSuccess;
}

private void btnLogin_Click(object sender, RoutedEventArgs e)
{
 LoginEventArgs args = new LoginEventArgs();

 manager.Select("member_tb", 2, tbId.Text, tbPw.Text);

 if (App.DataSearchResult == true)
 {
 args.isSuccess = true;
 }

 if (args.isSuccess == true)
 {
 MessageBox.Show("로그인에 성공하셨습니다!");

 Uri uri = new Uri("/mainpage.xaml", UriKind.Relative);
 NavigationService.Navigate(uri);
 }
 else
 {
 MessageBox.Show("로그인에 실패하셨습니다.");
 }
}
}
```

# Advanced WPF Programming

DataBase 연동(MySQL)

```
class LoginMysql
{
 public void Initialize()
 {
 Debug.WriteLine("DataBase Initialize");

 string connectionPath = $"SERVER=localhost;DATABASE=wpf_data;UID=root;PASSWORD=1234";
 App.connection = new MySqlConnection(connectionPath);
 }

 public MySqlCommand CreateCommand(string query)
 {
 MySqlCommand command = new MySqlCommand(query, App.connection);
 return command;
 }

 public bool OpenMySqlConnection()
 {
 try
 {
 App.connection.Open();
 return true;
 }
 catch (MySqlException e)
 {
 switch (e.Number)
 {
 case 0:
 Debug.WriteLine("Unable to Connect to Server");
 break;
 case 1045:
 Debug.WriteLine("Please check your ID or PassWord");
 break;
 }
 return false;
 }
 }
}
```

- LoginMysql.cs

# Advanced WPF Programming

DataBase 연동(MySQL)

- LoginMysql.cs 계속

```
public bool CloseMySQLConnection()
{
 try
 {
 App.connection.Close();
 return true;
 }
 catch (MySqlException e)
 {
 Debug.WriteLine(e.Message);
 return false;
 }
}

// Queyr Executer(Insert, Delete, Update ...)
public void MySQLQueryExecuter(string userQuery)
{
 string query = userQuery;

 if (OpenMySQLConnection() == true)
 {
 MySqlCommand command = new MySqlCommand(query, App.connection);

 if (command.ExecuteNonQuery() == 1)
 {
 Debug.WriteLine("값 저장 성공");
 App.DataSaveResult = true;
 }
 else
 {
 Debug.WriteLine("값 저장 실패");
 App.DataSaveResult = false;
 }

 CloseMySQLConnection();
 }
}
```

# Advanced WPF Programming

DataBase 연동(MySQL)

- LoginMysql.cs 계속

```
public List<string>[] Select(string tableName, int columnCnt, string id, string pw)
{
 string query = "SELECT id, pw FROM " + " " + tableName;

 List<string>[] element = new List<string>[columnCnt];

 for (int index = 0; index < element.Length; index++)
 {
 element[index] = new List<string>();
 }

 if (this.OpenMySqlConnection() == true)
 {
 MySqlCommand command = CreateCommand(query);
 MySqlDataReader dataReader = command.ExecuteReader();

 while (dataReader.Read())
 {
 element[0].Add(dataReader["id"].ToString());
 element[1].Add(dataReader["pw"].ToString());
 }
 }
}
```

# Advanced WPF Programming

DataBase 연동(MySQL)

- LoginMysql.cs 계속

```
 if (element != null)
 {
 for (int i = 0; i < element[0].Count; i++)
 {
 if (element[0][i].Contains(id))
 {
 for (int j = 0; j < element[1].Count; j++)
 {
 if (element[1][j].Contains(pw))
 {
 App.DataSearchResult = true;
 break;
 }
 }
 }
 break;
 }
 }

 dataReader.Close();
 this.CloseMySQLConnection();

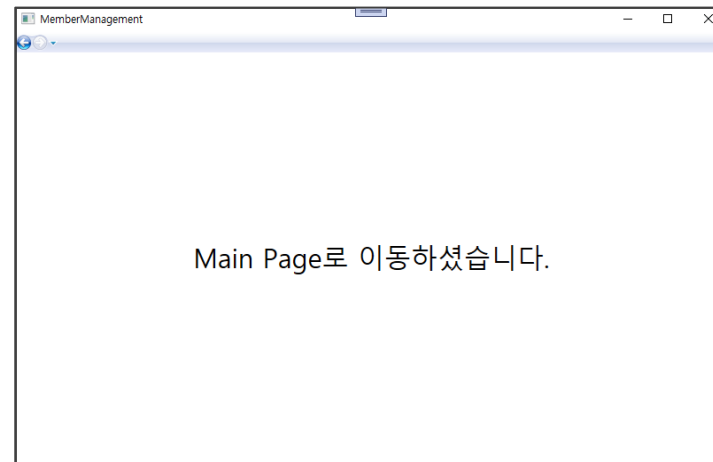
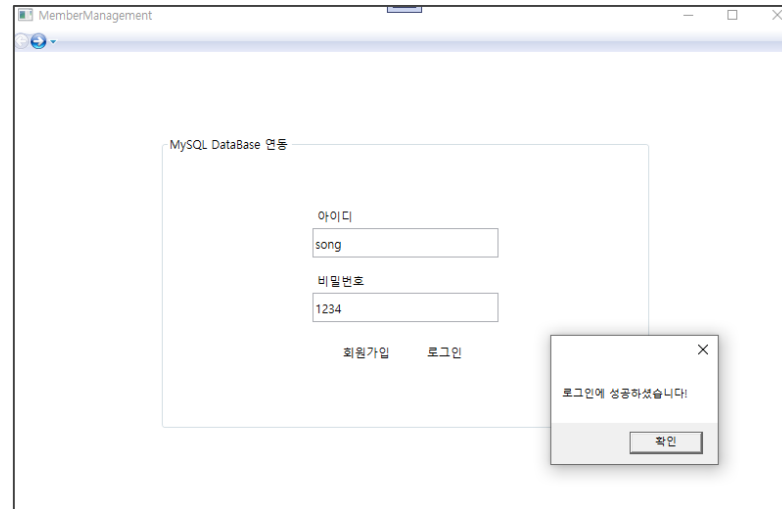
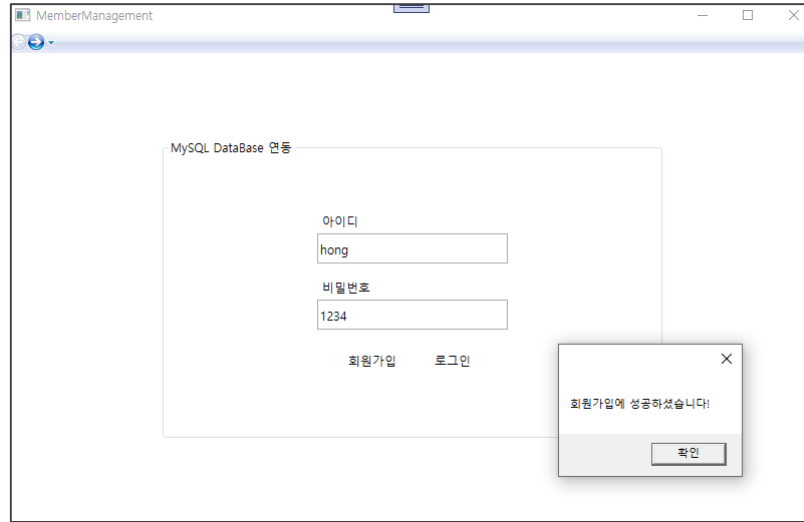
 return element;
 }
 else
 {
 return null;
 }
}
```



# Advanced WPF Programming

DataBase 연동(MySQL)

- 실행 결과



# Advanced WPF Programming

DataBase 설계 및 UI

- 실행 결과

# Advanced WPF Programming

DataBase 설계 및 UI

- 실행 결과

# Advanced WPF Programming

DataBase 설계 및 UI

- 실행 결과

# Advanced WPF Programming

DataBase 설계 및 UI

- 실행 결과

[C# 기반]

Advanced WPF Programming

**WPF 데이터 시각화**

# Advanced WPF Programming

## WPF 데이터 시각화

- WPF에서 데이터를 이용하여 Chart로 시각화 할 수 있는 라이브러리

라이브러리	지원 차트 유형	주요 특징		
LiveChart	선형, 막대, 원형, 산점도, 꺾은선 등	실시간 데이터, 애니메이션, MVVM 지원	실시간 데이터 시각화, 애니메이션, 다양한 차트 유형 지원	대량 데이터 처리에서 성능 저하 가능
OxyPlot	선형, 막대, 원형, 산점도, 히스토그램 등	가볍고 빠른 렌더링, 다양한 출력 포맷 지원	가벼운 성능, 빠른 처리, 다양한 포맷 내 보내기	고급 스타일링 및 애니메이션 부족
ScottPlot	선형, 막대, 산점도, 히스토그램, 히트맵	간단한 API, 빠른 성능, 상호작용 지원	대규모 데이터 처리에 최적화, 상호작용 기능	스타일링 및 애니메이션 제한적
Plotly.NET	선형, 막대, 원형, 산점도, 3D 등	상호작용 가능한 웹 기반 차트, 3D 차트 지원	웹 기반 상호작용 차트, 다양한 차트 및 스타일링 옵션	WPF에서 느릴 수 있음, 복잡할 수 있음
Modern UI Charts	선형, 막대, 원형 등	Modern UI 스타일, 간단한 차트 구현 가능	간단한 차트 시각화에 적합, Modern UI 스타일 적용 가능	복잡한 차트 기능 부족
Syncfusion Essential Studio for WPF(sfChart)	선형, 막대, 파이, 영역, 스플라인 등	대규모 데이터 처리, 실시간 업데이트, 3D 차트 지원	다양한 차트 유형 제공, 실시간 데이터 처리, 상호 작용 기능 제공, 비 상업용 무제 제공 등	초기 설정 복잡성, 패키지 크기가 큼(불필요한 기능도 포함)
Chart FX for WPF	선형, 막대, 파이, 금융 등	실시간 데이터, 고급 시각화 및 상호작용 기능	다양한 차트 유형 제공, 실시간 데이터 처리, 상호 작용 기능 제공 등	복잡한 설정, 커뮤니티 지원 부족

# Advanced WPF Programming

## WPF 데이터 시각화

### 1. LiveCharts 사용하기

- 무료 라이브러리
- <https://livecharts.dev/>
- 라이브러리 설치
  - NuGet 패키지 관리자 열기: Visual Studio에서 프로젝트를 열고, 도구 -> NuGet 패키지 관리자 -> 솔루션 Nuget 패키지 관리자 선택 -> LiveCharts 패키지 검색 및 설치
- XAML에 LiveCharts 추가
  - `xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"`



# Advanced WPF Programming

## WPF 데이터 시각화

- LiveChart 사용하기
  - 프로젝트 생성(WpfLiveChart) 후 라이브러리 설치

The screenshot displays the Visual Studio interface during the setup of a WPF application. On the left, the NuGet package manager window is open, showing a list of packages. The package **LiveCharts.Wpf** is highlighted with a red box, indicating it is the selected package for installation. The package details for **LiveCharts.Wpf** are shown on the right, including its version (0.9.7) and description. A dialog box titled "변경 내용 미리 보기" (Preview Changes) is also visible, showing the changes that will be made to the project when the package is installed. On the right side of the image, the project structure is shown in the Solution Explorer. The **WpfLiveChart** project is selected, and the **References** folder is expanded, showing a list of references. The **LiveCharts** and **LiveCharts.Wpf** packages are highlighted with a red box, indicating they are the installed packages.

# Advanced WPF Programming

## WPF 데이터 시각화

```
<Window x:Class="WpfLiveChart.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:WpfLiveChart"
 xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
 mc:Ignorable="d"
 Title="MainWindow" Height="450" Width="800">

 <Grid>
 <!-- 차트 컨트롤 추가 -->
 <lvc:CartesianChart Name="productChart" Series="{Binding ProductCollection}" Margin="10">
 <!-- X축 설정 -->
 <lvc:CartesianChart.AxisX>
 <lvc:Axis Title="X-Axis" Labels="{Binding Labels}"/>
 </lvc:CartesianChart.AxisX>
 <!-- Y축 설정 -->
 <lvc:CartesianChart.AxisY>
 <lvc:Axis Title="Y-Axis"/>
 </lvc:CartesianChart.AxisY>
 </lvc:CartesianChart>
 </Grid>
</Window>
```

# Advanced WPF Programming

## WPF 데이터 시각화

```
public partial class MainWindow : Window
{
 public SeriesCollection ProductCollection { get; set; }
 public string[] Labels { get; set; }

 public MainWindow()
 {
 InitializeComponent();

 ProductCollection = new SeriesCollection
 {
 new LineSeries
 {
 Title = "2023",
 Values = new ChartValues<double> { 10, 50, 39, 50, 42, 33 }
 },
 new LineSeries
 {
 Title = "2024",
 Values = new ChartValues<double> { 11, 56, 42, 48, 48, 43 }
 }
 };

 Labels = new[] { "Jan", "Feb", "Mar", "Apr", "May", "Jun" };
 DataContext = this;
 }
}
```

# Advanced WPF Programming

## WPF 데이터 시각화

- 다음과 같은 구조로 변경도 가능

```
public SeriesCollection Product { get; set; }
LineSeries series1;
LineSeries series2;
public string[] Labels { get; set; }
```

```
Product = new SeriesCollection();
```

```
series1 = new LineSeries
{
 Title = "2023",
 Values = new ChartValues<double> { 10, 50, 39, 50, 42, 33 }
};

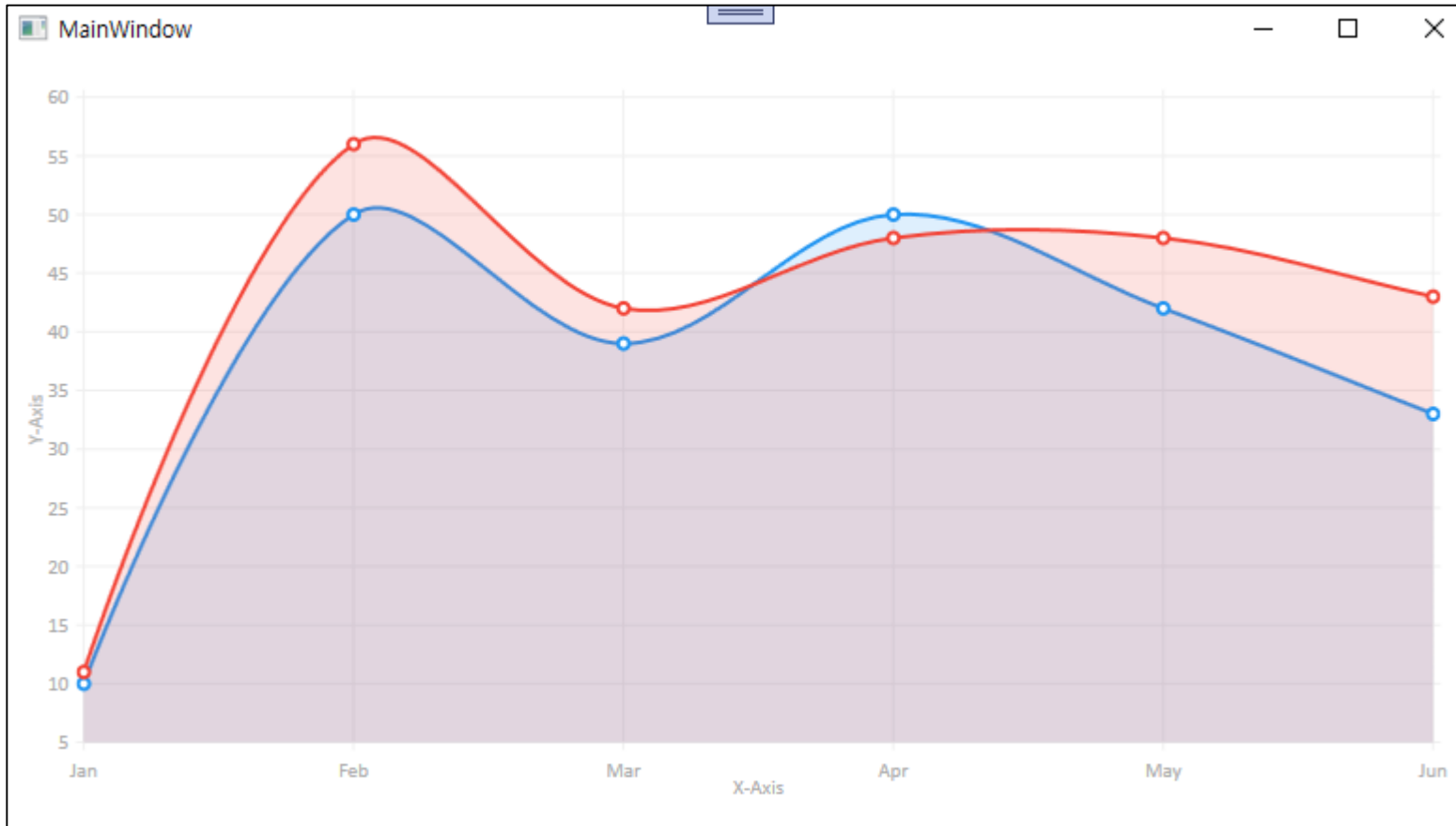
series2 = new LineSeries
{
 Title = "2024",
 Values = new ChartValues<double> { 11, 56, 42, 48, 48, 43 }
};

Product.Add(series1);
Product.Add(series2);
```

# Advanced WPF Programming

## WPF 데이터 시각화

- 결과



# Advanced WPF Programming

## WPF 데이터 시각화

- 차트 요소 설명
  - SeriesCollection: 차트에 표시될 데이터의 컬렉션입니다.
    - 각 데이터 시리즈는 다양한 차트 유형으로 표현 가능합니다.
      - LineSeries: 선형 차트
      - ColumnSeries: 막대 차트
      - PieSeries: 파이 차트
      - RowSeries: 행형 차트
      - ScatterSeries: 산점도
  - ChartValues: 차트에 시각화할 값들의 집합입니다.
  - Axis: X축과 Y축을 설정하는 데 사용됩니다.
    - Labels 속성을 통해 X축의 범주형 데이터를 설정할 수 있습니다.
  - DataContext: XAML의 바인딩을 위해 C# 코드의 데이터를 XAML에서 사용할 수 있도록 설정합니다.

# Advanced WPF Programming

## WPF 데이터 시각화

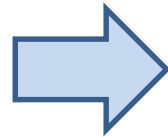
### a. LiveCharts 기능 이용

- LineSeries를 ColumnSeries로 변경하면 막대 모양의 Chart 결과를 얻을 수 있음

```
LineSeries series1;
LineSeries series2;
```

```
ColumnSeries series1;
ColumnSeries series2;
```

```
series1 = new LineSeries
{
 ...
};
```



```
series1 = new ColumnSeries
{
 ...
};
```

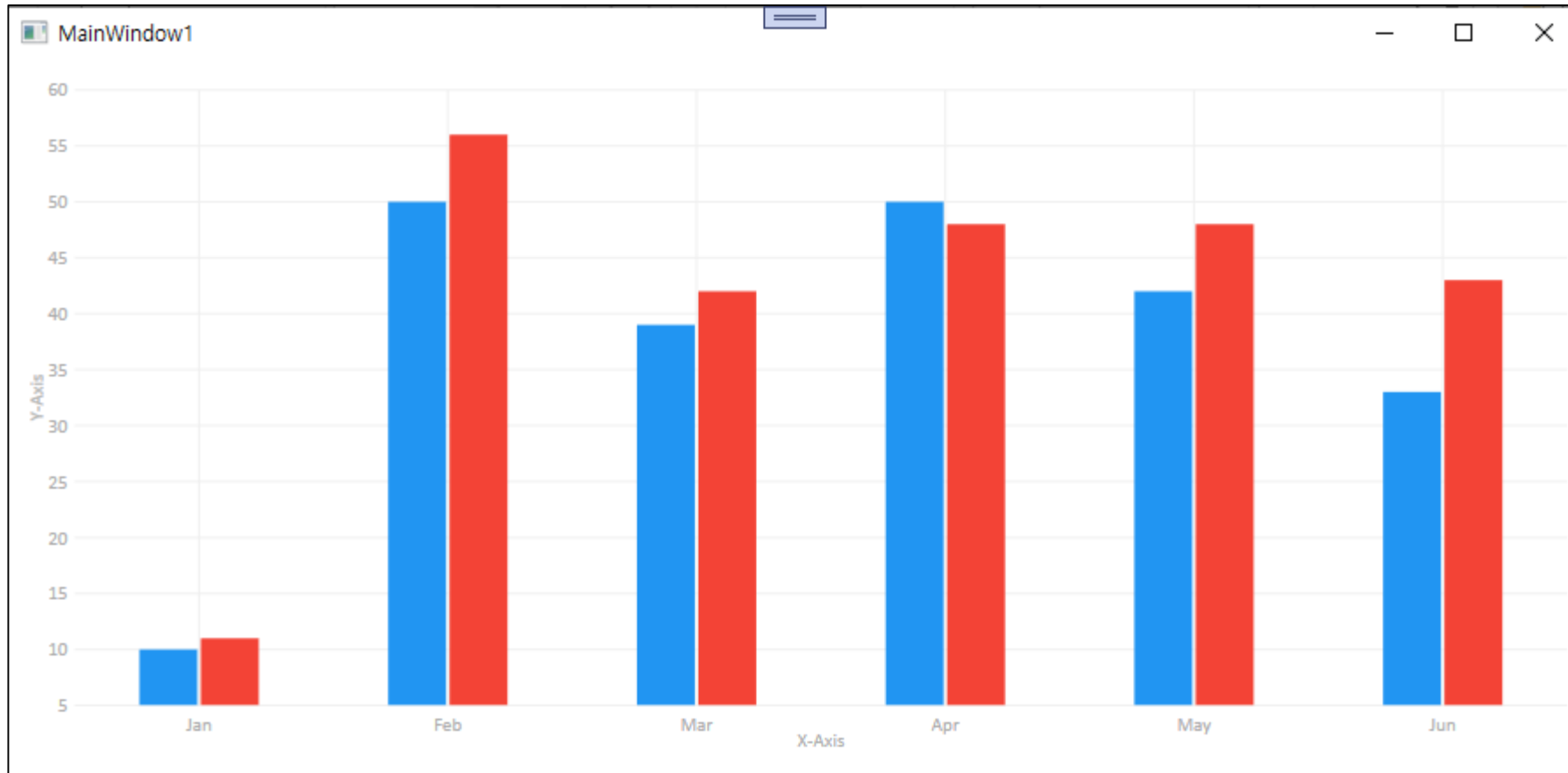
```
series2 = new LineSeries
{
 ...
};
```

```
series2 = new ColumnSeries
{
 ...
};
```

# Advanced WPF Programming

## WPF 데이터 시각화

- ColumnSeries Chart 결과





# Advanced WPF Programming

## WPF 데이터 시각화

### b. DataGrid Control과 LiveCharts 연동

- 지금까지의 코드는 Code-behind에서 데이터를 초기화 했었지만 XAML의 DataGrid에 차트에 표현할 데이터가 있는 경우로 변경
- 시나리오
  - DataGrid의 데이터 중 Category는 차트에 표현되는 부분이 아니며 Value 만 가능
  - 숫자로 표현된 Value 부분을 드래그하면 선택된 영역이 차트에 나타남
  - 드래그를 다시 하게 되면 차트는 항상 실시간으로 Update됨

Category	Value
A	10
B	20
C	30
D	40

# Advanced WPF Programming

## WPF 데이터 시각화

- XAML 코드에 LiveCharts.Wpf 등록

```
<Window x:Class="WpfLiveChart.DataGridChart"
 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:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"
 xmlns:local="clr-namespace:WpfLiveChart"
 mc:Ignorable="d"
 Title="DataGridChart" Height="450" Width="800">

 ...

</Window>
```

# Advanced WPF Programming

## WPF 데이터 시각화

```
<Grid>
 <Grid.ColumnDefinitions>
 <ColumnDefinition Width="2*" />
 <ColumnDefinition Width="3*" />
 </Grid.ColumnDefinitions>

 <DataGrid x:Name="DataGridExample"
 SelectionMode="Extended"
 SelectionUnit="Cell"
 AutoGenerateColumns="False"
 Grid.Column="0"
 ItemsSource="{Binding Data}">
 <DataGrid.Columns>
 <DataGridTextColumn Header="Category" Binding="{Binding Category}" />
 <DataGridTextColumn Header="Value" Binding="{Binding Value}" />
 </DataGrid.Columns>
 </DataGrid>

 <Ivc:CartesianChart x:Name="LiveChart" Grid.Column="1" Margin="10">
 <Ivc:CartesianChart.Series>
 <Ivc:ColumnSeries Values="{Binding ChartValues}" Title="Selected Data"/>
 </Ivc:CartesianChart.Series>
 </Ivc:CartesianChart>
</Grid>
```

← DataGrid 표현

← Chart 표현

# Advanced WPF Programming

## WPF 데이터 시각화

- Code-behind 처리

- 데이터 클래스

```
public class DataItem
{
 public string Category { get; set; }
 public double Value { get; set; }
}
```

- 데이터 저장할 Collection, 값 그리고 X축 Label 저장할 객체 선언

```
public ObservableCollection<DataItem> Data { get; set; }
public ChartValues<double> ChartValues { get; set; }
public List<string> Xlabel;
```

# Advanced WPF Programming

## WPF 데이터 시각화

- DataGrid에 초기화 할 값 생성
- XAML과 데이터 바인딩을 위한 설정
- DataGrid 영역이 변경될 때 발생할 이벤트 처리 메소드 등록

```
public DataGridChart()
{
 InitializeComponent();

 Data = new ObservableCollection<DataItem>
 {
 new DataItem { Category = "A", Value = 10 },
 new DataItem { Category = "B", Value = 20 },
 new DataItem { Category = "C", Value = 30 },
 new DataItem { Category = "D", Value = 40 }
 };

 ChartValues = new ChartValues<double>();

 DataContext = this;

 DataGridExample.SelectedCellsChanged += DataGridExample_SelectedCellsChanged;
}
```

# Advanced WPF Programming

## WPF 데이터 시각화

- DataGrid에서 선택된 셀의 값을 차트에 반영하는 함수

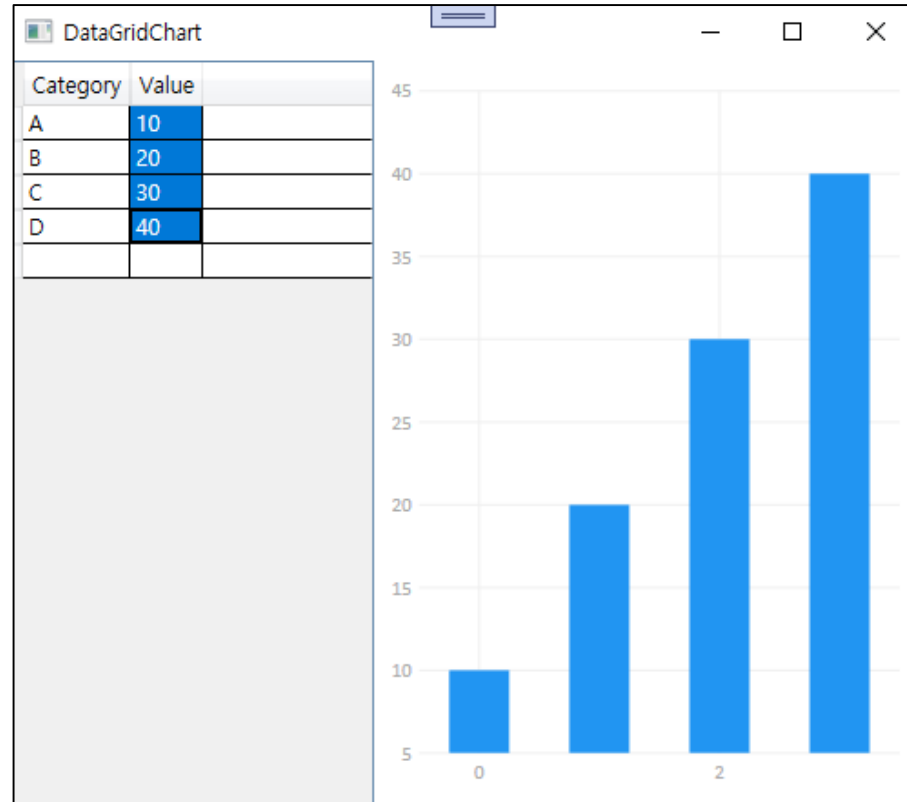
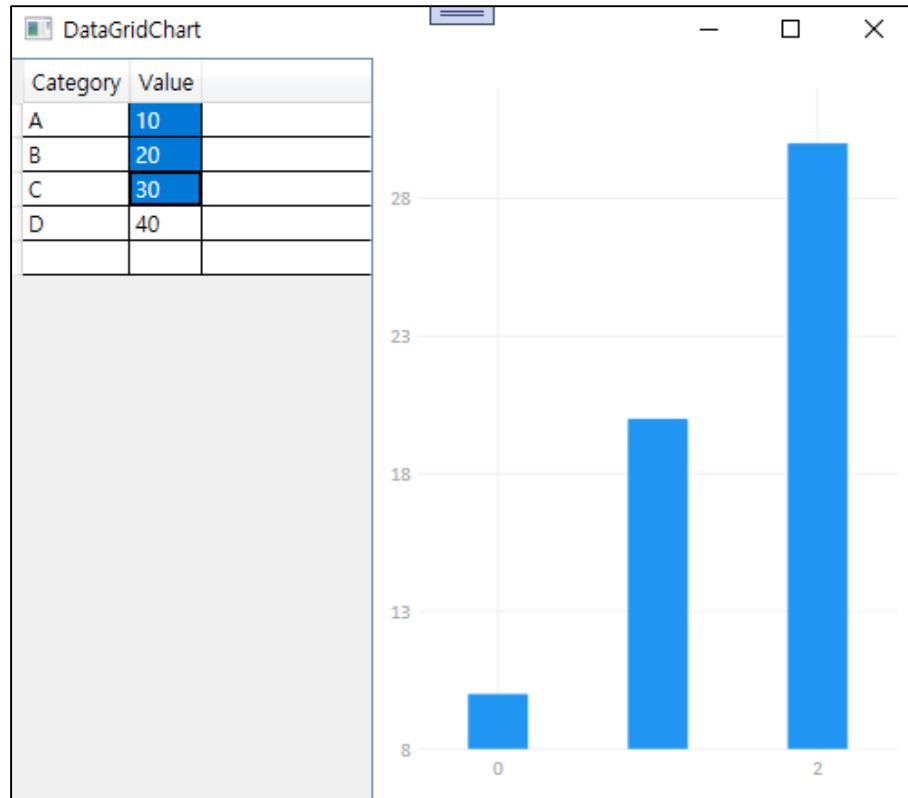
```
private void DataGridExample_SelectedCellsChanged(object sender, System.Windows.Controls.SelectedCellsChangedEventArgs e)
{
 ChartValues.Clear();

 foreach (var cell in DataGridExample.SelectedCells)
 {
 if (cell.Column.DisplayIndex == 1)
 {
 var value = ((DataItem)cell.Item).Value;
 ChartValues.Add(value);
 }
 }
}
```

- 두 번째 열(Values)에서만 데이터를 차트에 반영  
if (cell.Column.DisplayIndex == 1) ...

## WPF 데이터 시각화

- 선택 영역에 따른 Charts 결과
  - 드래그 영역이 달라지면 실시간으로 Charts가 Update 됨



# Advanced WPF Programming

## WPF 데이터 시각화

- LiveCharts의 기타 기능
  - 차트의 용도는 쓰임새마다 다르므로 필요한 기능은 제공된 LiveCharts 문서 및 예제를 참고할 수 있다.
  - 이 전 Charts에서도 여러가지 기능들을 생각하고 추가할 수 있다.
    - Charts 모양 변경
    - Charts에 표현될 데이터가 여러 개의 Series로 표현
    - 범례 추가
    - ...



# Advanced WPF Programming

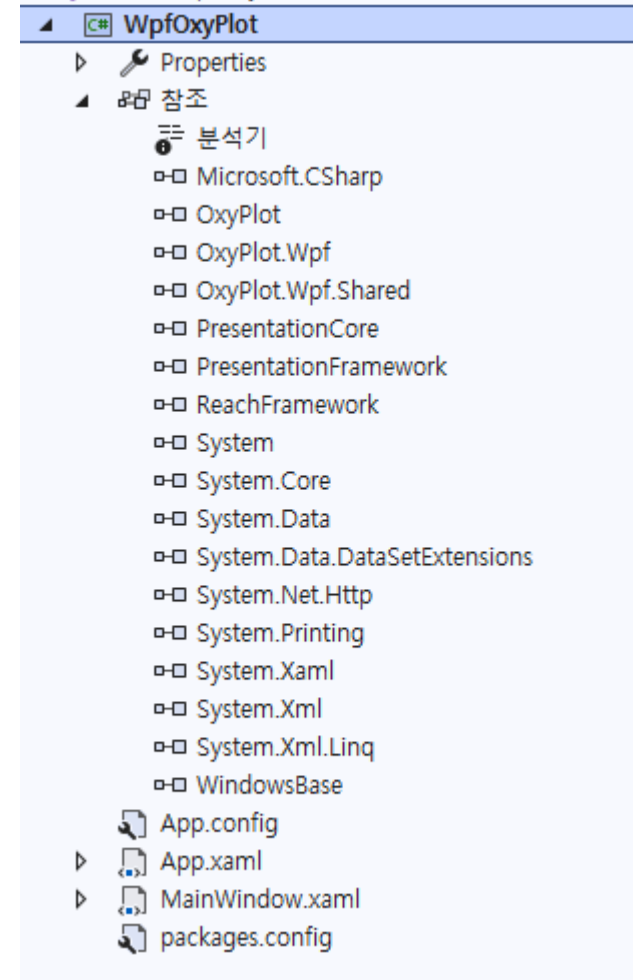
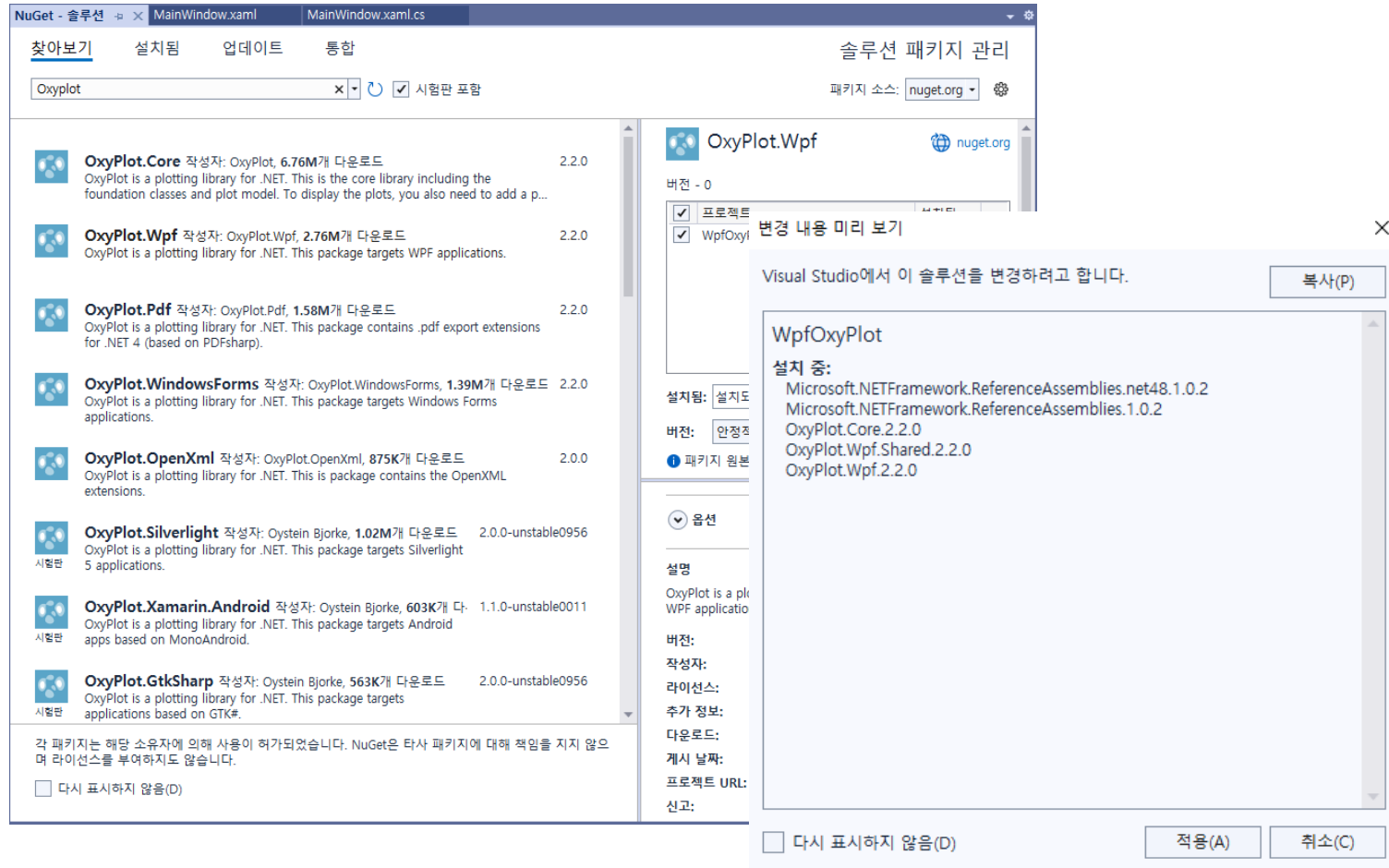
## WPF 데이터 시각화

### 2. OxyPlot 사용하기

- 무료 라이브러리
- <http://ww01.oxyplot.org/>
- 라이브러리 설치
  - NuGet 패키지 관리자 열기: Visual Studio에서 프로젝트를 열고, 도구 -> NuGet 패키지 관리자 -> 솔루션 Nuget 패키지 관리자 선택 -> OxyPlot.Wpf 패키지 검색 및 설치
- XAML에 LiveCharts 추가
  - `xmlns:oxy="http://oxyplot.org/wpf"`

# Advanced WPF Programming

## WPF 데이터 시각화



# Advanced WPF Programming

## WPF 데이터 시각화

```
<Window x:Class="WpfOxyPlot.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:WpfOxyPlot"
 xmlns:oxy="http://oxyplot.org/wpf"
 mc:Ignorable="d"
 Title="MainWindow" Height="450" Width="800">
 <Grid>
 <!-- PlotView: OxyPlot의 차트를 표시하는 컨트롤 -->
 <oxy:PlotView x:Name="plotView"
 Model="{Binding PlotModel}"
 HorizontalAlignment="Stretch"
 VerticalAlignment="Stretch"/>
 </Grid>
</Window>
```

# Advanced WPF Programming

## WPF 데이터 시각화

- XAML 설정
  - OxyPlot을 사용하기 위해 라이브러리 설정
    - `xmlns:lvc="clr-namespace:LiveCharts.Wpf;assembly=LiveCharts.Wpf"`
  - OxyPlot을 표현하기 위해 `<oxy:PlotView>`에 데이터 바인딩하고 기타 속성 설정
    - Model 속성에 필요한 데이터 바인딩

```
<oxy:PlotView x:Name="plotView"
 Model="{Binding PlotModel}"
 HorizontalAlignment="Stretch"
 VerticalAlignment="Stretch"/>
```

# Advanced WPF Programming

## WPF 데이터 시각화

```
public partial class MainWindow : Window
{
 public PlotModel PlotModel { get; set; }

 public MainWindow()
 {
 InitializeComponent();

 // PlotModel 초기화 및 데이터 추가
 PlotModel = new PlotModel { Title = "OxyPlot Example" };
 var series = new LineSeries
 {
 Title = "Sample Data",
 MarkerType = MarkerType.Circle
 };

 // 데이터를 시리즈에 추가
 series.Points.Add(new DataPoint(0, 0));
 series.Points.Add(new DataPoint(5, 18));
 series.Points.Add(new DataPoint(10, 12));
 series.Points.Add(new DataPoint(15, 25));
 series.Points.Add(new DataPoint(20, 28));
 }
}
```

# Advanced WPF Programming

## WPF 데이터 시각화

```
// PlotModel에 시리즈 추가
PlotModel.Series.Add(series);

// 축 추가
PlotModel.Axes.Add(new OxyPlot.Axes.LinearAxis {
 Position = OxyPlot.Axes.AxisPosition.Bottom, Title = "시간"
});
PlotModel.Axes.Add(new OxyPlot.Axes.LinearAxis {
 Position = OxyPlot.Axes.AxisPosition.Left, Title = "득점"
});

// DataContext 설정
DataContext = this;
}

}
```

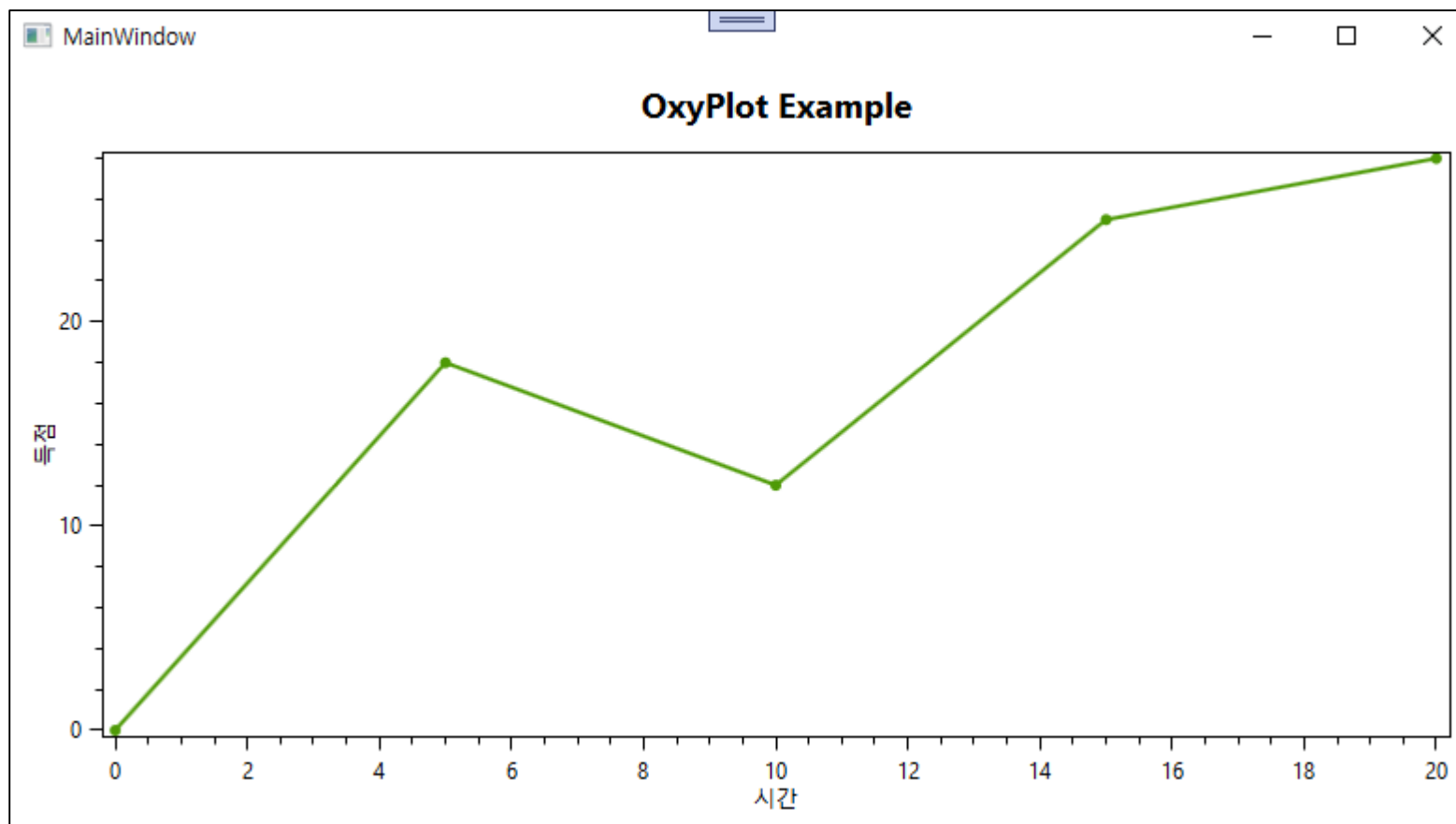
# Advanced WPF Programming

## WPF 데이터 시각화

- OxyPlot 차트 기본 요소 설명
  - PlotModel: 차트를 그리기 위한 모델입니다.
    - X축, Y축, 차트 데이터 시리즈 등을 관리합니다.
  - CategoryAxis: X축을 범주형 축으로 설정했습니다.
    - 각 데이터 포인트가 Category 1, Category 2 등의 범주를 나타냅니다.
  - LinearAxis: Y축을 선형 축으로 설정했습니다.
    - 차트에 표시되는 값의 범위를 0에서 100까지로 설정했습니다.
  - LineSeries: 첫 번째 시리즈는 선형 차트로 설정되어 있습니다.
    - 이 시리즈는 각 Category에 대해 선으로 연결된 데이터 포인트를 표시합니다.
  - DataPoint: LineSeries에서 각 데이터 포인트를 추가할 때 사용하는 객체로, X축과 Y축의 좌표를 입력합니다.

# Advanced WPF Programming

## WPF 데이터 시각화

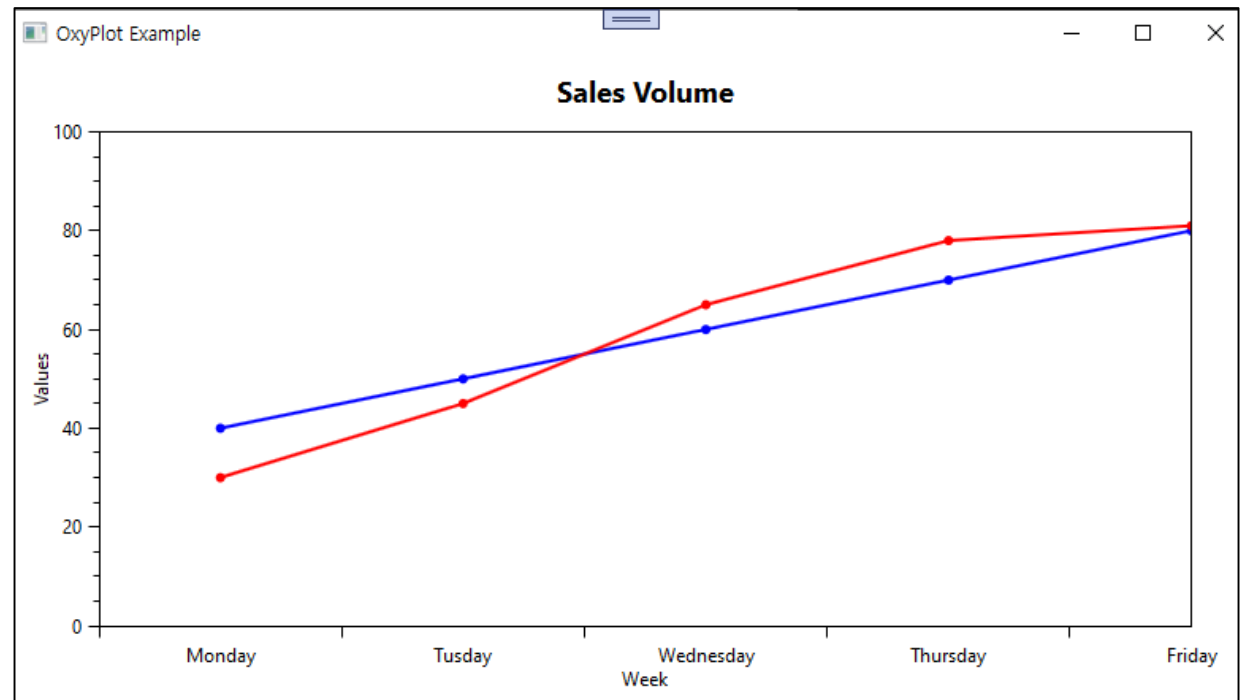




# Advanced WPF Programming

## WPF 데이터 시각화

- 여러 가지 데이터를 OxyPlot 이용하여 표현
  - OxyPlotWindow1 윈도우 생성
  - 추가 할 속성
    - X 축 Label로는 요일을 이용
    - 2가지 제품에 대한 판매량으로 가정
    - Values는 각 요일 별 판매량으로 가정
    - 2가지 제품을 서로 다른 색상으로 표현



# Advanced WPF Programming

## WPF 데이터 시각화

- 여러 가지 데이터를 OxyPlot 이용하여 표현

(a) OxyPlotWindow1.xaml 파일

- xmlns:oxy=<http://oxyplot.org/wpf> 추가
- Chart 표현 코드 추가

```
<oxy:PlotView x:Name="plotView"
 Model="{Binding SalesModel}"
 HorizontalAlignment="Stretch"
 VerticalAlignment="Stretch"/>
```

- Model 속성에 바인딩할 Model 지정

# Advanced WPF Programming

## WPF 데이터 시각화

- 여러 가지 데이터를 OxyPlot 이용하여 표현

(a) OxyPlotWindow1.xaml.cs 파일

- PlotModel Property 선언
  - `public PlotModel SalesModel { get; set; }`
- SalesModel 초기화
  - `SalesModel = new PlotModel{  
    Title = "Sales Volume"  
}`

# Advanced WPF Programming

## WPF 데이터 시각화

- 여러 가지 데이터를 OxyPlot 이용하여 표현

### (a) OxyPlotWindow1.xaml.cs 파일

- X축과 Y축 설정

```
SalesModel.Axes.Add(new CategoryAxis
{
 Position = AxisPosition.Bottom,
 Key = "CategoryAxis",
 Title = "Week",
 ItemsSource = new[] { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday" }
});

SalesModel.Axes.Add(new LinearAxis
{
 Position = AxisPosition.Left,
 Title = "Values",
 Minimum = 0,
 Maximum = 100
});
```

# Advanced WPF Programming

## WPF 데이터 시각화

- 여러 가지 데이터를 OxyPlot 이용하여 표현

(a) OxyPlotWindow1.xaml.cs 파일

- 첫번째 데이터 시리즈 생성

```
var lineSeries = new LineSeries
{
 Title = "TV",
 MarkerType = MarkerType.Circle,
 Color = OxyColors.Blue
};

lineSeries.Points.Add(new DataPoint(0, 40));
lineSeries.Points.Add(new DataPoint(1, 50));
lineSeries.Points.Add(new DataPoint(2, 60));
lineSeries.Points.Add(new DataPoint(3, 70));
lineSeries.Points.Add(new DataPoint(4, 80));
```

# Advanced WPF Programming

## WPF 데이터 시각화

- 여러 가지 데이터를 OxyPlot 이용하여 표현

(a) OxyPlotWindow1.xaml.cs 파일

- 두번째 데이터 시리즈 생성

```
var lineSeries1 = new LineSeries
{
 Title = "Phone",
 MarkerType = MarkerType.Circle,
 Color=OxyColors.Red
};

lineSeries1.Points.Add(new DataPoint(0, 30));
lineSeries1.Points.Add(new DataPoint(1, 45));
lineSeries1.Points.Add(new DataPoint(2, 65));
lineSeries1.Points.Add(new DataPoint(3, 78));
lineSeries1.Points.Add(new DataPoint(4, 81));
```

# Advanced WPF Programming

## WPF 데이터 시각화

- 여러 가지 데이터를 OxyPlot 이용하여 표현

(a) OxyPlotWindow1.xaml.cs 파일

- 시리즈 추가 및 DataContext 설정

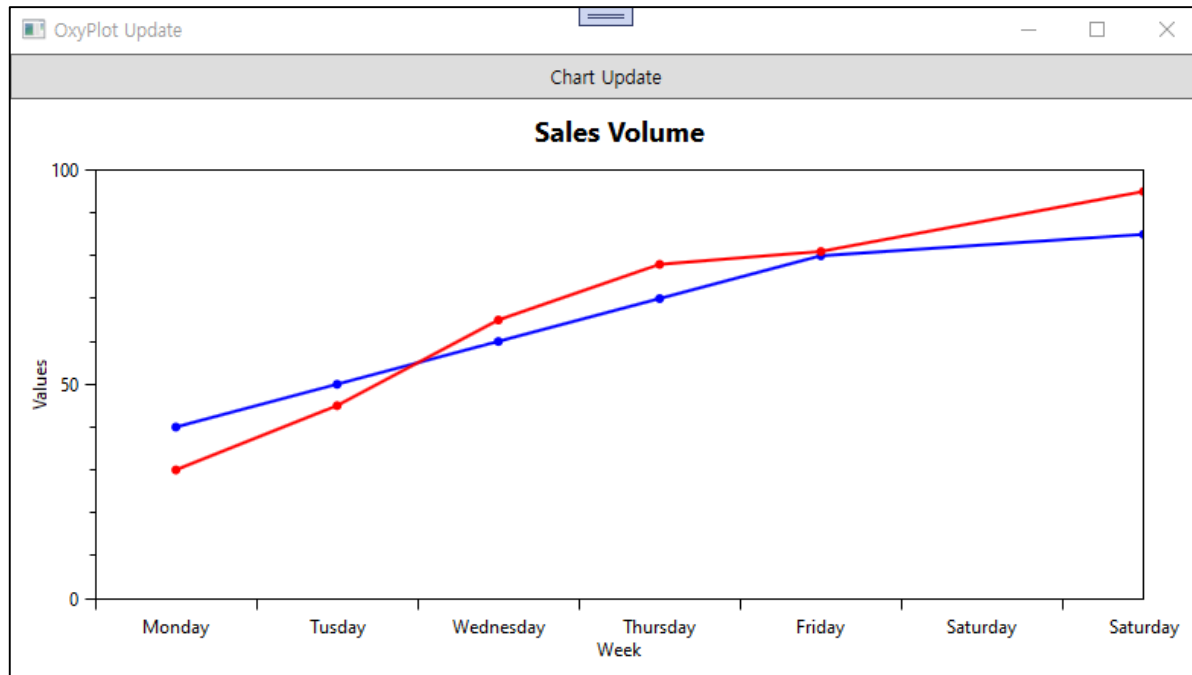
```
SalesModel.Series.Add(lineSeries);
SalesModel.Series.Add(lineSeries1);
```

```
DataContext = this;
```

# Advanced WPF Programming

## WPF 데이터 시각화

- OxyPlot에 새로운 값이 추가되어 실시간 Chart에 적용되는 경우
  - Update 시킬 버튼 컨트롤 추가
    - 기존 데이터 이외에 2개의 그래프에 값 추가
    - 추가된 값을 설명하는 X축 Label 추가





# Advanced WPF Programming

## WPF 데이터 시각화

- OxyPlot에 새로운 값이 추가되어 실시간 Chart에 적용되는 경우
  - XAML 파일에 Update 시킬 버튼 컨트롤 추가

```
<Grid>
 <Grid.RowDefinitions>
 <RowDefinition Height="30" />
 <RowDefinition />
 </Grid.RowDefinitions>

 <Button Grid.Row="0" Content="Chart Update" Click="Update" />

 <!-- PlotView를 이용하여 차트를 표시 -->
 <oxy:PlotView x:Name="plotView" Grid.Row="1"
 Model="{Binding SalesModel}"
 HorizontalAlignment="Stretch"
 VerticalAlignment="Stretch" />
</Grid>
```

# Advanced WPF Programming

## WPF 데이터 시각화

- OxyPlot에 새로운 값이 추가되어 실시간 Chart에 적용되는 경우
  - Code-behind 수정
    - Model은 전역에 배치되었으므로 그대로 이용
    - 데이터 추가에 따른 X축 Label을 업데이트 하기 위해 CategoryAxis를 전역으로 배치
    - 추가된 하루 동안 비교하려는 판매량을 담고 있는 LineSeries1, LineSeries2를 전역으로 배치

```
public PlotModel SalesModel { get; set; }
private CategoryAxis categoryAxis;
private LineSeries lineSeries1;
private LineSeries lineSeries2;
```

# Advanced WPF Programming

## WPF 데이터 시각화

- X축과 Y축 Label 추가

```
categoryAxis = new CategoryAxis()
{
 Position = AxisPosition.Bottom,
 Key = "CategoryAxis",
 Title = "Week",
 ItemsSource = new List<string> { "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday" }
};

var yAxis = new LinearAxis
{
 Position = AxisPosition.Left,
 Title = "Values",
 Minimum = 0,
 Maximum = 100
};

SalesModel.Axes.Add(categoryAxis);
SalesModel.Axes.Add(yAxis);
```

# Advanced WPF Programming

## WPF 데이터 시각화

- DataPoint 추가 및 차트 및 Chart Update 요청

```
private void Update(object sender, RoutedEventArgs e)
{
```

```
 var lineSeries = SalesModel.Series[0] as LineSeries;
 if (lineSeries != null)
 {
 lineSeries.Points.Add(new DataPoint(5, 85));
 }
```

← TV 판매량 추가

```
 var LineSeries1 = SalesModel.Series[1] as LineSeries;
 if (LineSeries1 != null)
 {
 LineSeries1.Points.Add(new DataPoint(5, 95));
 }
```

← Phone 판매량 추가

```
 var categories = categoryAxis.ItemsSource as List<string>;
 categories.Add("Saturday");
 categoryAxis.ItemsSource = categories;
```

← X축 Category 추가

```
 SalesModel.InvalidatePlot(true);
```

← 변경된 Chart Update

```
}
```

# Advanced WPF Programming

## WPF 데이터 시각화

- OxyPlot Chart에 범례 추가
  - Legend: Legend 객체를 생성하여 범례의 속성을 설정합니다.
    - LegendPlacement: 범례가 차트 안에 위치할지, 차트 밖에 위치할지를 설정합니다.
      - Outside로 설정하면 차트 외부에 표시됩니다.
    - LegendPosition: 범례의 위치를 설정합니다.
      - TopRight는 차트의 우상단에 배치합니다.
    - LegendOrientation: 범례의 항목을 세로(Vertical) 또는 가로(Horizontal)로 나열할지를 설정합니다.
    - LegendBorderThickness: 범례 테두리의 두께를 설정합니다.
    - LegendSymbolLength: 범례에서 각 시리즈를 나타내는 기호의 길이를 설정합니다.
    - PlotModel.Legends.Add(legend): PlotModel에 범례 객체를 추가합니다.

# Advanced WPF Programming

## WPF 데이터 시각화

- OxyPlot Chart에 범례 추가 코드

```
var legend = new Legend
{
 LegendPlacement = LegendPlacement.Outside,
 LegendPosition = LegendPosition.TopRight,
 LegendOrientation = LegendOrientation.Vertical,
 LegendBorderThickness = 1,
 LegendSymbolLength = 24
};

SalesModel.Legends.Add(legend);
```

# Advanced WPF Programming

시각화 설계

# Advanced WPF Programming

시각화 설계



# Advanced WPF Programming

시각화 설계

[C# 기반]

## Advanced WPF Programming

WPF 프로젝트 주제 및 설계

# Advanced WPF Programming

## 프로젝트 설계

- 만들어 볼 내용 구상
- UI 구상
- MVVM으로 프로젝트 생성 및 환경 준비
- 필요한 라이브러리 결정

# Advanced WPF Programming

프로젝트 설계

# Advanced WPF Programming

프로젝트 설계

# Advanced WPF Programming

프로젝트 설계

# Advanced WPF Programming

프로젝트 설계

# Advanced WPF Programming

프로젝트 설계



# Advanced WPF Programming

프로젝트 설계