**01장 Application과 Window**

**1) GrowAndShrink**

|  |
| --- |
| using System;  using System.Windows;  using System.Windows.Input;  namespace Petzold.GrowAndShrink  {  public class GrowAndShrink : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new GrowAndShrink());  }  public GrowAndShrink()  {  Title = "Grow & Shrink";  WindowStartupLocation = WindowStartupLocation.CenterScreen;  Width = 192;  Height = 192;  }  protected override void OnKeyDown(KeyEventArgs args)  {  base.OnKeyDown(args);  if (args.Key == Key.Up)  {  //Left -= 0.05 \* Width;  //Top -= 0.05 \* Height;  Width \*= 1.1;  Height \*= 1.1;  }  else if (args.Key == Key.Down)  {  Left += 0.05 \* (Width /= 1.1);  Top += 0.05 \* (Height /= 1.1);  }  }  }  } |

예)1

예 1번은 Window 객체를 상속받은 GrowAndShrink클래스의 표준 어플리케이션 창을 띄우고 버튼 다운 이벤트를 이용하여 창의 크기를 조절하는 예제이다 상속받은 FrameworkElement의 클래스의 요소들을 이용하여 값(Title , Width, Height)을 변경 할 수 있다.

WPF 프로젝트의 예

//MainWindow.xaml

<Window x:Class="문학수.MainWindow"

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

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

Title="MainWindow" Height="350" Width="525" KeyDown="OnKeyDown">

<Grid>

</Grid>

</Window>

WPF 코드에선 xaml 코드에서 키 다운 이벤트를 주고

//MainWindow.xaml.cs

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void OnKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Up)

{

Width \*= 1.1;

Height \*= 1.1;

}

else if(e.Key == Key.Down)

{

Left += 0.05 \* (Width /= 1.1);

Top += 0.05 \* (Height /= 1.1);

}

}

}

CS 코드에서 xaml 에서 정의한 이벤트를 처리 한다.

**2) HandleAnEvent**

|  |
| --- |
| using System;  using System.Windows;  using System.Windows.Input;  namespace Petzold.HandleAnEvent  {  class HandleAnEvent  {  [STAThread]  public static void Main()  {  Application app = new Application();  Window win = new Window();  win.Title = "Handle An Event";  win.MouseDown += WindowOnMouseDown;  app.Run(win);  }  static void WindowOnMouseDown(object sender, MouseButtonEventArgs args)  {  //Window win = sender as Window;  Window win = Application.Current.MainWindow;  string strMessage =  string.Format("Window clicked with {0} button at point ({1})",  args.ChangedButton, args.GetPosition(win));  MessageBox.Show(strMessage, win.Title);  }  }  } |

예)2

예2 번은 마우스 다운 이벤트를 지정하고 이벤트가 발생 할 때에 좌표값을 불러와 메시지 박스에 출력 하는 예제 이다.

Window 객체를 생성하고 현제 Mainwindow를 할당 한 뒤

이벤트가 발생 되면 메시지 박스로 MainWindow 의 이벤트발생 지점의 마우스 좌표 값을불러와 메시지 박스로 출력 하여 주는 것이다.

다음 WPF 프로젝트의 소스를 보자

WPF 프로젝트의 예

//MainWindow.xaml

<Window x:Class="WpfApplication2.MainWindow"

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

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

Name="qwe"

Title="MainWindow" Height="350" Width="525" ResizeMode="CanResizeWithGrip" WindowStartupLocation="Manual" MouseUp="MouseUpbutton" >

<Grid>

</Grid>

</Window>

WPF 프로젝트에선 xaml 코드와 C# 코드가 분리 되어있다

Xaml 코드에서 Mainwindow를 불러올 때 Mainwindow객체의 Name을 정의 한뒤 Mainwindow에서 이벤트를 등록 시킨다.

이벤트를 등록 시킨뒤에

//MainWindow.xaml.cs

namespace WpfApplication2

{

/// <summary>

/// MainWindow.xaml에 대한 상호 작용 논리

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void MouseUpbutton(object sender, MouseButtonEventArgs e)

{

Point pos = e.GetPosition(qwe);

MessageBox.Show(pos.ToString());

}

}

}

CS 에서 해당 이벤트의 대한 정의를 내려주면 된다.

**3) InheritAppAndWindow**

|  |
| --- |
| using System;  using System.Windows;  using System.Windows.Input;  namespace Petzold.InheritAppAndWindow  {  class InheritAppAndWindow  {  [STAThread]  public static void Main()  {  MyApplication app = new MyApplication();  app.Run();  }  }  } |

예)3

|  |
| --- |
| using System;  using System.Windows;  using System.Windows.Input;  namespace Petzold.InheritAppAndWindow  {  class MyApplication : Application  {  protected override void OnStartup(StartupEventArgs args)  {  base.OnStartup(args);  MyWindow win = new MyWindow();  win.Show();  }  }  } |

예)4

|  |
| --- |
| using System;  using System.Windows;  using System.Windows.Input;  namespace Petzold.InheritAppAndWindow  {  public class MyWindow : Window  {  public MyWindow()  {  Title = "Inherit App & Window";  }  protected override void OnMouseDown(MouseButtonEventArgs args)  {  base.OnMouseDown(args);  string strMessage =  string.Format("Window clicked with {0} button at point ({1})",  args.ChangedButton, args.GetPosition(this));  MessageBox.Show(strMessage, Title);  }  }  } |

예)5

다음 예제 3,4,5 는 콘솔에서 MainWindow 가 실행되는 과정이다

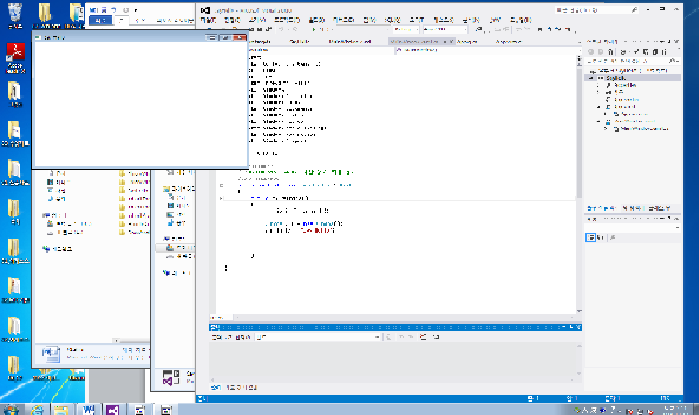
InheritAppAndWindow 에서 App와Window를 상속 받겠다는 클래스를 지정한다.

MyApplication 의 객체를 생성한뒤 MyWinow의 객체를 생성한다.

MyWindow 객체를 생성한뒤 실행 순서에 따라 show() 를 만나면서 메인 윈도우가 실행된다.

**4) SayHello**

|  |
| --- |
| using System;  using System.Windows;  namespace Petzold.SayHello  {  class SayHello  {  [STAThread]  public static void Main()  {  Window win = new Window();  win.Title = "Say Hello";  win.Show();  Application app = new Application();  app.Run();  }  }  } |

****

**PPT 01 참조 [STAThread]**

- **Window** 객체는 표준 어플리케이션 창을 생성하고자 할 때 사용한다.

- **Title** 속성은 생성되는 창의 제목 표시줄에 나타날 글자를 지정하고

**Show** 함수는 화면에 창을 표시한다.

**-WPF(SayHello)**

<Window x:Class="SayHello.MainWindow"

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

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

Title="Say Hello" Height="350" Width="525">

<Grid>

</Grid>

</Window>

namespaceSayHello

**{**

publicpartialclassMainWindow **:** Window

**{**

publicMainWindow**()**

**{**

InitializeComponent**();**

Windowwin=newWindow**();**

win.Title="Say Hello"**;**

**}**

**}**

**}**

**5) ) InheritTheApp**

|  |
| --- |
| using System;  using System.Windows;  using System.Windows.Input;  namespace Petzold.InheritTheApp  {  class InheritTheApp : Application  {  [STAThread]  public static void Main(string[] args)  {  InheritTheApp app = new InheritTheApp();  app.Run();  }  protected override void OnStartup(StartupEventArgs args)  {  base.OnStartup(args);  Window win = new Window();  win.Title = "Inherit the App";  win.Show();  }  protected override void OnSessionEnding(SessionEndingCancelEventArgs args)  {  base.OnSessionEnding(args);  MessageBoxResult result =  MessageBox.Show("Do you want to save your data?",  MainWindow.Title, MessageBoxButton.YesNoCancel,  MessageBoxImage.Question, MessageBoxResult.Yes);  args.Cancel = (result == MessageBoxResult.Cancel);  }  }  }**C:\Users\53\Desktop\InheritTheApp.png** |

**- Application** 객체에는 **OnStartup** 함수는 **Run**을 호출한 직후 호출되는 함수로 **Window** 객체를 생성해서 화면에 출력해준다. 때문에 재정의를 통해 **Window** 객체를 생성할 수 있다.

**- Application** 객체에는 **OnSessionEnding** 함수를 재정의를 통해 사용자의 종료 이벤트가 발생하면 메시지 박스를 생성하여 결과를 리턴 받아 **SessionEndingCancelEventArgs** 객체를 통해 윈도우가 종료되거나 로그오프가 되는 것을 막는다.

**-WPF(InheritTheApp)**

<Window x:Class="InheritTheApp.MainWindow"

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

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

Title="Inherit the App" Height="350" Width="525" Closing="Window\_Closing\_1">

<Grid>

</Grid>

</Window>

namespaceInheritTheApp

**{**

publicpartialclassMainWindow **:** Window

**{**

publicMainWindow**()**

**{**

InitializeComponent**();**

**}**

privatevoidWindow\_Closing\_1**(**objectsender**,** System.ComponentModel.CancelEventArgse**)**

**{**

MessageBoxResultresult=

MessageBox.Show**(**"Do you want to save your data?"**,** "종료하시겠습니까?"**,** MessageBoxButton.YesNoCancel**,**

MessageBoxImage.Question**,** MessageBoxResult.Yes**);**

if **(**result==MessageBoxResult.Cancel**)**

**{**

e.Cancel= **(**result==MessageBoxResult.Cancel**);**

**}**

elseif **(**result==MessageBoxResult.No**)**

**{**

e.Cancel= **(**result==MessageBoxResult.No**);**

**}**

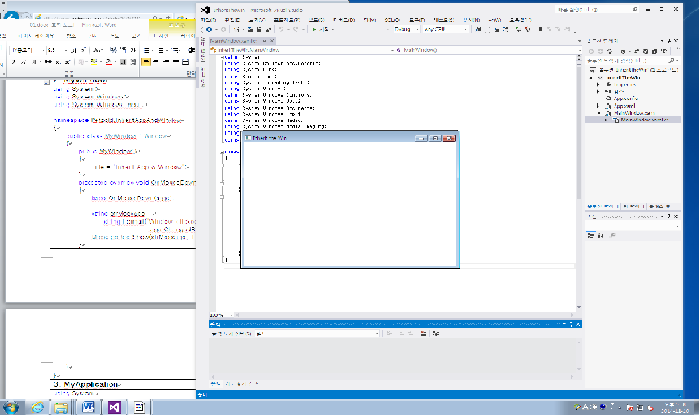
**}**

**}**

**}**

**6) InheritTheWin**

|  |
| --- |
| using System;  using System.Windows;  using System.Windows.Input;  namespace Petzold.InheritTheWin  {  class InheritTheWin : Window  {  [STAThread]  public static void Main()  {    Application app = new Application();  app.Run(new InheritTheWin());  }  public InheritTheWin()  {  Left = 500;  Top = 250;    Left = SystemParameters.PrimaryScreenHeight - Width;  Top = SystemParameters.PrimaryScreenHeight - Height;  Left = (SystemParameters.WorkArea.Width - Width) / 2 +  SystemParameters.WorkArea.Left;  Top = (SystemParameters.WorkArea.Height - Height) / 2 +  SystemParameters.WorkArea.Top;  //사각형을 정중앙에 놓기 위해 모니터 작업영역에 사각형의 너비를 뺀것의 1/2에 사각형의 왼쪽x좌표  를 더했다. 높이도 마찬가지.  Title = "Inherit the Win";  }  }  } |



**-WPF(InheritTheWin)**

<Window x:Class="InheritTheWin.MainWindow"

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

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

Title="MainWindow" Height="350" Width="525">

<Grid>

</Grid>

</Window>

namespaceInheritTheWin

**{**

/// <summary>

/// MainWindow.xaml에 대한 상호 작용 논리

/// </summary>

publicpartialclassMainWindow **:** Window

**{**

publicMainWindow**()**

**{**

InitializeComponent**();**

this.Left= **(**SystemParameters.WorkArea.Width-Width**)** /2+

SystemParameters.WorkArea.Left**;**

this.Top= **(**SystemParameters.WorkArea.Height-Height**)** /2+

SystemParameters.WorkArea.Top**;**

Title="Inherit the Win"**;**

**}**

**}**

**}**

**7) ThrowWindowParty**

|  |
| --- |
| using System;  using System.Windows;  using System.Windows.Input;  namespace Petzold.ThrowWindowParty  {  class ThrowWindowParty : Application  {  [STAThread]  public static void Main()  {  ThrowWindowParty app = new ThrowWindowParty();  app.Run();  }  protected override void OnStartup(StartupEventArgs args)  {  Window winMain = new Window();  winMain.Title = "Main Window";  winMain.MouseDown += WindowOnMouseDown;  winMain.Show();  for (int i = 0; i < 2; i++)  {  Window win = new Window();  win.Title = "Extra Window No. " + (i + 1);  win.Show();  }  }  void WindowOnMouseDown(object sender, MouseButtonEventArgs args)  {  Window win = new Window();  win.Title = "Modal Dialog Box";  win.ShowDialog();  }  }  } |

**1. 여러 종류의 Windows**

**- OnStartup** 함수의 재정의를 통해 윈도우를 생성하지만 모두가 동등한 기능을 가지고 있기 때문에 기능만으로는 메인 윈도우 창을 확인 할 수 없다. 다만 맨 처음 **Show** 함수를 **winMain**객체가 호출하기 때문에 메인 창으로 간주 할 수 있을 것이다.

**- Application** 클래스에는 **Windows** 속성값을 통해 여러 개의 **Window** 객체를 저장한다. 때문에 여기서 **Windows[0]**이 **“Main Window”**이란 제목의 창을 가리키고 있는 것을 확인 할 수 있다.

**-** 생성한 **Window** 객체 **ShowInTaskbar**의 속성 값을 **false**로 수정하면 작업 표시줄에 보이지 않는다.

- 생성한 **Window** 객체의 **ShowDialog** 함수를 호출하면 모달 대화상자가 된다.

**2. 메인 Window**

- **Application** 객체의 **ShutdownMode**의 속성값은 열거형의 맴버로 지정되어 있다. 기본으로는 **OnLastWindowClose**가 지정되어 마지막 윈도우 창이 종료되면 **Run**을 반환하고 종료한다. 이것을 **OnMainWindowClose**를 지정하면 메인 윈도우 창이 종료되면 **Run**을 반환하고 종료한다.

- **MainWindow**의 속성값에 생성한 **Window** 객체를 대입한 객체가 메인 윈도우 창으로 지정이 된다. 하지만 **ShutdownMode**를 지정해야 한다.

- **ShutdownMode**를 지정하지 않고 메인 윈도우 창을 종료하는 방법은 메인 윈도우 창을 이후 생성되는 객체의 **Owner**의 속성값에 메인 창으로 설정할 **Window** 객체 대입하여 부모와 자식관계를 구축 할 수 있다. 이 관계가 구축되면 부모 창을 종료하면 부모 창이 소유하고 있는 자식 창들이 모두 소멸한다. 즉, 모달리스 대화상자가 되는 것이다.

WPF 프로젝트의 예

-여기서 기존의 OnStarup(App\_Startup 으로 사용) 이벤트를 xmal에서 등록을 시키자.

|  |
| --- |
| //App.xmal  <Application x:Class="ThrowWindowParty.App"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Startup="App\_Startup">  <Application.Resources>    </Application.Resources>  </Application>  <!-- StartupUri="MainWindow.xaml" 가 있으면 메인 윈도우 xaml도생성.--> |

-그 후 App\_Startup 이벤트 함수를 생성하자.

|  |
| --- |
| //App.xmal.cs  using System;  using System.Windows;  using System.Windows.Input;  namespace ThrowWindowParty  {  /// <summary>  /// App.xaml에 대한 상호 작용 논리  /// </summary>  public partial class App : Application  {  protected void App\_Startup(object sender, StartupEventArgs args)  {  Window winMain = new Window();  winMain.Title = "Main Window"; //타이틀 표현  winMain.MouseDown += WindowOnMouseDown; //이벤트 등록  winMain.Show(); //출력  for (int i = 0; i < 2; i++)  {  Window win = new Window(); //윈도우 생성  win.Title = "Extra Window No. " + (i + 1); //타이틀 이름  win.Show(); //모달리스 출력  win.Owner = winMain; //소유자 / 소유관계 설정  winMain = win; //메인 윈도우 설정  }  }  void WindowOnMouseDown(object sender, MouseButtonEventArgs args)  {  Window win = new Window();  win.Title = "Modal Dialog Box";  win.ShowDialog();  }  }  } |

**8) ThrowWindowParty2**

**1.** 7) 예제에서는 Extra Window No.2 가 MainWindow였었다. 이번에는 이것을 풀고 테스트를해보자.

-1. 위와 같은 예제처럼 Startup의 이벤트를 등록하자.

|  |
| --- |
| //App.xmal  <Application x:Class="ThrowWindowParty.App"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Startup="App\_Startup">  <Application.Resources>    </Application.Resources>  </Application>  <!-- StartupUri="MainWindow.xaml" 가 있으면 메인 윈도우 xaml도생성.--> |

-2.

|  |
| --- |
| //App.xmal.cs  using System;  using System.Windows;  using System.Windows.Input;  namespace ThrowWindowParty2  {  /// <summary>  /// App.xaml에 대한 상호 작용 논리  /// </summary>  public partial class App : Application  {  private void App\_Startup(object sender, StartupEventArgs e)  {  Window winMain = new Window();  winMain.Title = "Main Window";  winMain.MouseDown += WindowOnMouseDown;  winMain.Show();  for (int i = 0; i < 2; i++)  {  Window win = new Window();  win.Title = "Extra Window No. " + (i + 1);  win.Show();//모달리스 출력  win.Owner = winMain; //소유자 / 소유관계 설정  }  }  void WindowOnMouseDown(object sender, MouseButtonEventArgs args)  {  Window win = new Window();  win.Title = "Modal Dialog Box";  win.ShowDialog();  }  }  } |

**9) TypeYourTitle**

- PreviewTextInput, TextInput는 컨트롤이 텍스트 입력을 받았을 때 발생하게 된다. 다음 내용을 보자. 다음 예제는 텍스트를 입력받으면 윈도우 타이틀에 입력된 문자를 표시하는 예제이다.

1)TextInput의 이벤트를 등록하여보자.

|  |
| --- |
| //MainWIndow.xaml  <Window x:Class="TypeYourTitle.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Title="" Height="350" Width="525"  TextInput="TextInput1">  <Grid>    </Grid>  </Window> |

2)등록된 TextInput1 이벤트 함수를 구성하여 보자.

|  |
| --- |
| //MainWIndow.xaml.cs  using System;  using System.Windows;  using System.Windows.Input;  namespace TypeYourTitle  {  /// <summary>  /// MainWindow.xaml에 대한 상호 작용 논리  /// </summary>  public partial class MainWindow : Window  {  public MainWindow()  {  InitializeComponent();  }  private void TextInput1(Object sender, TextCompositionEventArgs e)  {  base.OnTextInput(e);  if (e.Text == "\b" && Title.Length > 0)  Title = Title.Substring(0, Title.Length - 1); //이 인스턴스에서 부분 문자열을 검색합니다.  else if (e.Text.Length > 0 && !Char.IsControl(e.Text[0]))  Title += e.Text;//그렇지 않은경우 타이틀 내용 추가.  }  }  } |