**01장 Application과 Window**

**1) 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]**

**1. Window 객체**

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

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

**Show** 함수는 화면에 창을 표시한다.

**2. Application 객체**

- **Application** 객체는 어플리케이션을 동작시키고자 할 때 사용한다.

- **Run** 함수를 통해 키보드나 마우스 등의 사용자 입력을 받을 수 있게 된다.

호출하지 않으면 그대로 Main함수가 종료 되는 시점에 객체가 소멸된다.

**PPT 02~04 참조 WPF의 클래스 계층도 And 구조**

**3. Window 객체 And Application 객체**

- **Window** 객체는 화면상에 표시 된다. 생성 개수의 제한은 없다.

- **Application** 객체는 화면상에 표시되지 않지만 실행된 프로그램에서 어느 부분에서도

접근이 가능한 전역 객체이다. 생성 개수는 한 개로 한 객체만 동작할 수 있다.

- **Window** 객체를 생성하기 전에 **Application** 객체의 **Run** 함수를 호출하면 이후 코드는

실행하지 않는다.

- **Application** 객체는 화면상에 표시되지 않지만 실행된 프로그램에서 어느 부분에서도

접근이 가능한 전역 객체이다.

**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;  string strMessage =  string.Format("Window clicked with {0} button at point ({1})",  args.ChangedButton, args.GetPosition(win));  MessageBox.Show(strMessage, win.Title);  }  }  } |

**1. 정적 이벤트 등록하기**

- **Window** 클래스는 이벤트를 정의한 **UIElement** 클래스를 상속 받아 정의 되어 있기 때문에

**Window** 객체에 있는 **MouseDown**이벤트에 **WindowOnMouseDown** 이벤트 헨들러를 대입하면

**MouseDown**이벤트가 발생 할 때마다 호출이 되어 화면상에 메시지 박스가 표시된다.

**- MouseButtonEventArgs** 클래스는 **Input** 네임스페이스에 정의 되어 있다.

이 클래스를 통해 생성된 인스턴스 객체의 **GetPosition** 함수에 인자로 **Window** 객체를

넘기면 클라이언트 영역의 좌측상단을 기준으로 마우스의 좌표를 반환한다.

**2. Application에 등록된 Window 객체**

**- Application** 객체에는 **Current** 속성에는 **MainWindow** 속성을 통해 **Window** 객체를 이벤트

헨들러에서 할당 할 수 있다.

**3) HandleAnEvent**

|  |
| --- |
| 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);  }  }  } |

**1. Application클래스를 상속 받아 재정의하는 객체의 이벤트 함수**

**- 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.

**4) 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.

**5) InheritAppAndWindow**

**1. 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();  }  }  } |

**2. MyWindow**

|  |
| --- |
| 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);  }  }  } |

**3. MyApplication**

|  |
| --- |
| 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();  }  }  } |

**PPT 05 참조 장치 독립적인 단위**

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