19장 XAML

**XAML(Extensible Application Markup Language)**

정의

XAML은 선언적 태그 언어입니다. XAML은 .NET Framework 프로그래밍 모델에 적용되어 .NET Framework 응용 프로그램 UI를 쉽게 만들 수 있게 합니다. 선언적 XAML 태그에 표시되는 UI 요소를 만든 다음 코드 숨김 파일을 사용하여 UI 정의를 런타임 논리와 구분할 수 있습니다. 이 정의는 partial 클래스 정의를 통해 태그에 연결됩니다. XAML은 WPF의 프로그래밍 추가적인 인터페이스로 XML 코드도 역시 유효한 XAML이며 대부분 객체의 생성과 초기화를 위해 설계되었다. 다음 2가지 코드는 동일하게 대응된다.

|  |
| --- |
| 결국 XAML이란.  1. "XAML이란 객체들의 속성과 그 상관관계를 XML 기반의 언어로 표현하는 기법"  2. "Application에서 필요한 객체의 구조를 구성하는 언어" 입니다. |

Xml

|  |
| --- |
| <Button ForeGround = “LightSeaGreen” FontSize = “24pt”>  Hello XAML! //일반적으로 버튼 객체의 Content 프로퍼티에 입력되는 텍스트  </Button> |

C#코드

|  |
| --- |
| Button btn = new Button();  btn.Foreground = Brushes.LightSeaGreen;  btn.FontSize = 32;  btn.Content = "Hello, XAML!"; |

XAML을 사용하면 애플리케이션의 외형과 기능을 분리할 수 있고, 이벤트 핸들러는 일반적으로 C# 형태의 코드로 기술되지만 이벤트 핸들러를 XAML의 데이터 바인딩으로 대체하는 것도 가능하다.

XAML 문서를 명확하게 구분하기 위하여 XML 네임스페이스가 고안되었으며, 네임스페이스의 이름은 유일하고 영속적이므로 URL을 사용하는 것이 일반적이다. WPF 프로그램에서 사용하는 URL은 다음과 같다.

|  |
| --- |
| http://schemas.microsoft.com/winfx/2006/xaml/presentation |

|  |
| --- |
| <StackPanel xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">  <Button HorizontalAlignment="Center" Margin="24">  Hello, XAML!  </Button>  <Ellipse Width="200" Height="100" Margin="24"  Stroke="Red" StrokeThickness="10" />  <ListBox Width="100" Height="100" Margin="24">  <ListBoxItem>Sunday</ListBoxItem>  <ListBoxItem>Monday</ListBoxItem>  <ListBoxItem>Tuesday</ListBoxItem>  <ListBoxItem>Wednesday</ListBoxItem>  <ListBoxItem>Thursday</ListBoxItem>  <ListBoxItem>Friday</ListBoxItem>  <ListBoxItem>Saturday</ListBoxItem>  </ListBox>  </StackPanel> |

XML 파일은 하나의 루트 엔리먼트만을 가지는데, 이 파일에서는 StackPanel이 루트 엔리먼트이다.

WPF 프로그램은 XamlReader.Load을 사용해 XAML 덩어리를 객체로 전환한다. 루트 엘리먼트에 자식 엘리먼트가 있으면 그 엘리먼트도 마찬가지로 변환된다.

XamlReader.Load는 Stream 객체나 XmlReader 객체를 요구한다.

|  |
| --- |
| StringReader strreader = new StringReader(strXaml);  XmlTextReader xmlreader = new XmlTextReader(strreader);  object obj = XamlReader.Load(xmlreader); |

1번 예제 595p

strXaml 문자열을 정의하고 그 안에 XAML 문서를 객체로 변환한 후 그 객체를 Window의 Content 프로퍼티로 설정 하여 XAML문서의 내용으로 UI를 구현

|  |
| --- |
| //-------------------------------------------------  // LoadEmbeddedXaml.cs (c) 2006 by Charles Petzold  //-------------------------------------------------  using System;  using System.IO;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Markup;  using System.Xml;  namespace Petzold.LoadEmbeddedXaml  {  public class LoadEmbeddedXaml : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new LoadEmbeddedXaml());  }  public LoadEmbeddedXaml()  {  Title = "Load Embedded Xaml";  string strXaml =  "<Button xmlns='http://schemas.microsoft.com/" +  "winfx/2006/xaml/presentation'" +  " Foreground='LightSeaGreen' FontSize='24pt'>" +  "Click me!" +  "</Button>";  StringReader strreader = new StringReader(strXaml);  XmlTextReader xmlreader = new XmlTextReader(strreader);  object obj = XamlReader.Load(xmlreader);  Content = obj;  }  } |

2번 예제 597p

FindName메소드를 사용하여 엘리먼트 트리상에서 특정한 이름으로 엘리먼트를 찾아 버튼 이벤트 핸들러에 연결하는 코드로 실행 시 로드한 XAML에게 이벤트 핸들러를 연결 할 수 있는 가장 간단한 방법이다.

|  |
| --- |
| <StackPanel xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">  <Button Name="MyButton"  HorizontalAlignment="Center"  Margin="24">  Hello, XAML!  </Button>  <Ellipse Width="200"  Height="100"  Margin="24"  Stroke="Red"  StrokeThickness="10" />  <ListBox Width="100"  Height="100"  Margin="24">  <ListBoxItem>Sunday</ListBoxItem>  <ListBoxItem>Monday</ListBoxItem>  <ListBoxItem>Tuesday</ListBoxItem>  <ListBoxItem>Wednesday</ListBoxItem>  <ListBoxItem>Thursday</ListBoxItem>  <ListBoxItem>Friday</ListBoxItem>  <ListBoxItem>Saturday</ListBoxItem>  </ListBox>  </StackPanel> |

|  |
| --- |
| using System;  using System.IO;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Markup;  namespace Petzold.LoadXamlResource  {  public class LoadXamlResource : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new LoadXamlResource());  }  public LoadXamlResource()  {  Title = "Load Xaml Resource";  Uri uri = new Uri("pack://application:,,,/LoadXamlResource.xml");  Stream stream = Application.GetResourceStream(uri).Stream;  FrameworkElement el = XamlReader.Load(stream) as FrameworkElement;  Content = el;  Button btn = el.FindName("MyButton") as Button;  if (btn != null)  btn.Click += ButtonOnClick;  }  void ButtonOnClick(object sender, RoutedEventArgs args)  {  MessageBox.Show("The button labeled '" +  (args.Source as Button).Content + "' has been clicked");  }  } |

3번 예제 599p

XamlReader.Load의 리턴 값을 Window 객체로 변환하여, 버튼 클릭 이벤트 핸들러를 비주얼 트리속에서 찾아 연결하지 않고 직접 Window의 AddHandler를 호출해 연결하는 코드.

|  |
| --- |
| <!-- ================================================  LoadXamlWindow.xml (c) 2006 by Charles Petzold  ================================================ -->  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  Title="Load Xaml Window"  SizeToContent="WidthAndHeight"  ResizeMode="CanMinimize">  <StackPanel>  <Button HorizontalAlignment="Center"  Margin="24">  Hello, XAML!  </Button>  <Ellipse Width="200"  Height="100"  Margin="24"  Stroke="Red"  StrokeThickness="10" />  <ListBox Width="100"  Height="100"  Margin="24">  <ListBoxItem>Sunday</ListBoxItem>  <ListBoxItem>Monday</ListBoxItem>  <ListBoxItem>Tuesday</ListBoxItem>  <ListBoxItem>Wednesday</ListBoxItem>  <ListBoxItem>Thursday</ListBoxItem>  <ListBoxItem>Friday</ListBoxItem>  <ListBoxItem>Saturday</ListBoxItem>  </ListBox>  </StackPanel>  </Window> |

|  |
| --- |
| //-----------------------------------------------  // LoadXamlWindow.cs (c) 2006 by Charles Petzold  //-----------------------------------------------  using System;  using System.IO;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Markup;  namespace Petzold.LoadXamlWindow  {  public class LoadXamlWindow  {  [STAThread]  public static void Main()  {  Application app = new Application();  Uri uri = new Uri("pack://application:,,,/LoadXamlWindow.xml");  Stream stream = Application.GetResourceStream(uri).Stream;  Window win = XamlReader.Load(stream) as Window;  win.AddHandler(Button.ClickEvent,  new RoutedEventHandler(ButtonOnClick));  app.Run(win);  }  static void ButtonOnClick(object sender, RoutedEventArgs args)  {  MessageBox.Show("The button labeled '" +  (args.Source as Button).Content +  "' has been clicked");  }  } |

4번 예제 602p

Open File 대화상자를 사용해서 XAML 파일을 하드디스크에서 로드하는 프로그램. Window를 루트 엘리먼트로 가지고 있다면 윈도우를 띄우고 그렇지 않으면 Frame의 Content 프로퍼티로 설정 후 구현.

|  |
| --- |
| //---------------------------------------------  // LoadXamlFile.cs (c) 2006 by Charles Petzold  //---------------------------------------------  using Microsoft.Win32;  using System;  using System.IO;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Markup;  using System.Xml;  namespace Petzold.LoadXamlFile  {  public class LoadXamlFile : Window  {  Frame frame;  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new LoadXamlFile());  }  public LoadXamlFile()  {  Title = "Load XAML File";  DockPanel dock = new DockPanel();  Content = dock;  // Create button for Open File dialog.  Button btn = new Button();  btn.Content = "Open File...";  btn.Margin = new Thickness(12);  btn.HorizontalAlignment = HorizontalAlignment.Left;  btn.Click += ButtonOnClick;  dock.Children.Add(btn);  DockPanel.SetDock(btn, Dock.Top);  // Create Frame for hosting loaded XAML.  frame = new Frame();  dock.Children.Add(frame);  }  void ButtonOnClick(object sender, RoutedEventArgs args)  {  OpenFileDialog dlg = new OpenFileDialog();  dlg.Filter = "XAML Files (\*.xaml)|\*.xaml|All files (\*.\*)|\*.\*";  if ((bool)dlg.ShowDialog())  {  try  {  // Read file with XmlTextReader.  XmlTextReader xmlreader = new XmlTextReader(dlg.FileName);  // Convert XAML to object.  object obj = XamlReader.Load(xmlreader);  // If it's a Window, call Show.  if (obj is Window)  {  Window win = obj as Window;  win.Owner = this;  win.Show();  }  // Otherwise, set as Content of Frame.  else  frame.Content = obj;  }  catch (Exception exc)  {  MessageBox.Show(exc.Message, Title);  }  }  }  } |

605 – 606p

XAML 고유 엘리먼트나 속성을 사용하고 싶다면 두 번째 네임스페이스 선언을 XAML 파일에 포함시켜야 하는데, 그 선언은 다음 URL을 참조한다.

|  |
| --- |
| http://schemas.microsoft.com/winfx/2006/xaml |

XAML 네임스페이스는 관례적으로 접두어 x와 결합되고 Class 속성과 Code 엘리먼트는 XAML 파일에서 x:Class와 x:Code 같은 참조 방법으로 나타난다.

x:Class 속성은 Window 루트 엘리먼트 속에서 매우 자주 보이며, XAML 파일은 다음과 같은 전체적인 구조를 가지고 있다.

|  |
| --- |
| <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  x:Class="MyNamespace.MyClassName"  ….>  ….  </Window> |

|  |
| --- |
| Public namespace MyNamespace  {  Public partial class MyClassName : Window  {  …  }  } |

이것은 주로 이벤트 핸들러나 몇 가지 초기화 코드를 포함하는 코드 비하인드 파일이며, XAML 파일에서 정의된 컨트롤이나 엘리먼트를 지원한다.

Partial 키워드는 MyClassName 클래스가 다른 어딘가에 추가적인 코드를 가지고 있다는 것을 명시해 그 코드는 C# 코드 비하인드 파일 속에 있다.

5번 예제 607p

CompileXamlWindow 프로그램 속에 작성된 XAML과 C# 코든 같은 연결은 XAML을 코드와 함께 컴파일 했을때만 가능하다.

|  |
| --- |
| <!-- ====================================================  CompileXamlWindow.xaml (c) 2006 by Charles Petzold  ==================================================== -->  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  x:Class="Petzold.CompileXamlWindow.CompileXamlWindow"  Title="Compile XAML Window"  SizeToContent="WidthAndHeight"  ResizeMode="CanMinimize">  <StackPanel>  <Button HorizontalAlignment="Center"  Margin="24"  Click="ButtonOnClick">  Click the Button  </Button>  <Ellipse Name="elips"  Width="200"  Height="100"  Margin="24"  Stroke="Black"/>  <ListBox Name="lstbox"  Width="150"  Height="150"  Margin="24"  SelectionChanged="ListBoxOnSelection" />  </StackPanel>  </Window> |

CompileXamlWindow 클래스를 C#으로 코딩한 부분에서 이 필드를 바로 참조할 수 있는데, 다음이 CompileXamlWindow 클래스의 나머지 부분에 해당하는 코드 비하인드 파일이다.

|  |
| --- |
| //--------------------------------------------------  // CompileXamlWindow.cs (c) 2006 by Charles Petzold  //--------------------------------------------------  using System;  using System.Reflection;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Input;  using System.Windows.Media;  namespace CompileXamlWindow  {  public partial class CompileXamlWindow : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new CompileXamlWindow());  }  public CompileXamlWindow()  {  // Required method call to hook up event handlers and  // initialize fields.  InitializeComponent();  // Fill up the ListBox with brush names.  foreach (PropertyInfo prop in typeof(Brushes).GetProperties())  lstbox.Items.Add(prop.Name);  }  // Button event handler just displays MessageBox.  void ButtonOnClick(object sender, RoutedEventArgs args)  {  Button btn = sender as Button;  MessageBox.Show("The button labled '" + btn.Content +  "' has been clicked.");  }  // ListBox event handler changes Fill property of Ellipse.  void ListBoxOnSelection(object sender, SelectionChangedEventArgs args)  {  ListBox lstbox = sender as ListBox;  string strItem = lstbox.SelectedItem as string;  PropertyInfo prop = typeof(Brushes).GetProperty(strItem);  elips.Fill = (Brush)prop.GetValue(null, null);  }  } |

* 610쪽 4번째 문단

자신의 클래스를 XAML 내에서 사용하기 위해 C# 파일에 잇는 사용자 정의 클래스를 네임스페이스를 선언하기만 하면 된다.

|  |
| --- |
| Xmlns:stuff=”clr-namespace:MyNamespace” |

“clr-namespace” 문자열은 반드시 소문자여야 하고 콜론이 붙어야 한다.

XAML파일에서 ColorGridBox 클래스를 사용하기 위해서 다음과 같이 네임스페이스를 선언 할 필요가 있다.

|  |
| --- |
| Xmlns:src=”clr-namespace:Petzold.SelectColorFromGrid” |

6번 예제 612p

접두어 src를 사용해 ColorGridBox를 참조하는 네임스페이스를 포함하는 코드. Xmlns:src 로 clr-namespace를 정하였으면 이 클래스를 사용 할 때에는 src:(클래스명) 으로 작성 해야 한다.

|  |
| --- |
| <!-- =================================================  UseCustomClass.xaml (c) 2006 by Charles Petzold  ================================================= -->  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:src="clr-namespace:Petzold.SelectColorFromGrid"  x:Class="Petzold.UseCustomClass.UseCustomClass"  Title="Use Custom Class"  SizeToContent="WidthAndHeight"  ResizeMode="CanMinimize">  <StackPanel Orientation="Horizontal">  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24">  Do-nothing button to test tabbing  </Button>  <src:ColorGridBox HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24"  SelectionChanged="ColorGridBoxOnSelectionChanged" />  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24">  Do-nothing button to test tabbing  </Button>  </StackPanel>  </Window> |

|  |
| --- |
| //-----------------------------------------------  // UseCustomClass.cs (c) 2006 by Charles Petzold  //-----------------------------------------------  using SelectColorFromGrid;  using System;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Input;  using System.Windows.Media;  namespace Petzold.UseCustomClass  {  public partial class UseCustomClass : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new UseCustomClass());  }  public UseCustomClass()  {  InitializeComponent();  }  void ColorGridBoxOnSelectionChanged(object sender,  SelectionChangedEventArgs args)  {  ColorGridBox clrbox = args.Source as ColorGridBox;  Background = (Brush) clrbox.SelectedValue;  }  }  } |

7번 예제 614p

이 프로젝트는 Window로부터 파생된 클래스를 위해 별도의 XAML 파일을 포함하고 있는데, 이 파일은 XML의 네임스페이스에 프로젝트의 네임스페이스를 가지고 있기 때문에 StackPanel이 5개의 CenteredButton 클래스 인스턴스를 가질 수 있다.

|  |
| --- |
| <!-- =================================================  CenteredButton.xaml (c) 2006 by Charles Petzold  ================================================= -->  <Button xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  x:Class="Petzold.UseCustomXamlClass.CenteredButton"  HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="12" /> |

|  |
| --- |
| <!-- =====================================================  UseCustomXamlClass.xaml (c) 2006 by Charles Petzold  ===================================================== -->  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:src="clr-namespace:Petzold.UseCustomXamlClass"  x:Class="Petzold.UseCustomXamlClass.UseCustomXamlClass"  Title = "Use Custom XAML Class">  <StackPanel Name="stack">  <src:CenteredButton>Button A</src:CenteredButton>  <src:CenteredButton>Button B</src:CenteredButton>  <src:CenteredButton>Button C</src:CenteredButton>  <src:CenteredButton>Button D</src:CenteredButton>  <src:CenteredButton>Button E</src:CenteredButton>  </StackPanel>  </Window> |

|  |
| --- |
| //---------------------------------------------------  // UseCustomXamlClass.cs (c) 2006 by Charles Petzold  //---------------------------------------------------  using System;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Input;  using System.Windows.Media;  namespace Petzold.UseCustomXamlClass  {  public partial class UseCustomXamlClass : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new UseCustomXamlClass());  }  public UseCustomXamlClass()  {  InitializeComponent();  for (int i = 0; i < 5; i++)  {  CenteredButton btn = new CenteredButton();  btn.Content = "Button No. " + (i + 1);  stack.Children.Add(btn);  }  }  }  } |

8번 예제 617p

프로그램에 명확한 Main 메소드가 없는 프로젝트로 StartupUri 태그를 통해 프로그램이 시작 될 때 UI 리소스를 자동으로 로드.

|  |
| --- |
| <!-- ===========================================  MyWindow.xaml (c) 2006 by Charles Petzold  =========================================== -->  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  x:Class="Petzold.IncludeApplicationDefinition.MyWindow"  Title="Include Application Definition"  SizeToContent="WidthAndHeight"  ResizeMode="CanMinimize">  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="1.5in"  Click="ButtonOnClick">  Click the Button  </Button>  </Window> |
|  |

|  |
| --- |
| //-----------------------------------------  // MyWindow.cs (c) 2006 by Charles Petzold  //-----------------------------------------  using System;  using System.Windows;  using System.Windows.Controls;  using System.Windows.Input;  namespace Petzold.IncludeApplicationDefinition  {  public partial class MyWindow : Window  {  public MyWindow()  {  InitializeComponent();  }  void ButtonOnClick(object sender, RoutedEventArgs args)  {  Button btn = sender as Button;  MessageBox.Show("The button labled '" + btn.Content +  "' has been clicked.");  }  }  } |

|  |
| --- |
| <!-- ================================================  MyApplication.xaml (c) 2006 by Charles Petzold  ================================================ -->  <Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  x:Class="Petzold.IncludeApplicationDefinition.MyApplication"  StartupUri="MyWindow.xaml" /> |

* MyApplication.g.cs 파일에 Main 메소드가 있다.

9번 예제 620p

프로젝트에 코드 파일 없이(많은 어플리케이션에 해당되지 않음) 하나의 XAML 파일만 갖는 예제

|  |
| --- |
| <!-- =================================================  XamlOnlyWindow.xaml (c) 2006 by Charles Petzold  ================================================= -->  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  Title="Compile XAML Only"  SizeToContent="WidthAndHeight"  ResizeMode="CanMinimize">  <StackPanel>  <Button HorizontalAlignment="Center"  Margin="24">  Just a Button  </Button>  <Ellipse Width="200"  Height="100"  Margin="24"  Stroke="Red"  StrokeThickness="10" />  <ListBox Width="100"  Height="100"  Margin="24">  <ListBoxItem>Sunday</ListBoxItem>  <ListBoxItem>Monday</ListBoxItem>  <ListBoxItem>Tuesday</ListBoxItem>  <ListBoxItem>Wednesday</ListBoxItem>  <ListBoxItem>Thursday</ListBoxItem>  <ListBoxItem>Friday</ListBoxItem>  <ListBoxItem>Saturday</ListBoxItem>  </ListBox>  </StackPanel>  </Window> |

|  |
| --- |
| <!-- ==============================================  XamlOnlyApp.xaml (c) 2006 by Charles Petzold  ============================================== -->  <Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  StartupUri="XamlOnlyWindow.xaml" /> |

10번 예제 622p

XAML에 내장된 코드를 넣는 프로그램

|  |
| --- |
| <!-- ==================================================  EmbeddedCodeApp.xaml (c) 2006 by Charles Petzold  ================================================== -->  <Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  StartupUri="EmbeddedCodeWindow.xaml" /> |

|  |
| --- |
| <!-- =====================================================  EmbeddedCodeWindow.xaml (c) 2006 by Charles Petzold  ===================================================== -->  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  x:Class="Petzold.CompileXamlOnly.EmbeddedCodeWindow"  Title="Embed Code in XAML"  SizeToContent="WidthAndHeight"  ResizeMode="CanMinimize"  Loaded="WindowOnLoaded">  <StackPanel>  <Button HorizontalAlignment="Center"  Margin="24"  Click="ButtonOnClick">  Click the Button  </Button>  <Ellipse Name="elips"  Width="200"  Height="100"  Margin="24"  Stroke="Red"  StrokeThickness="10" />  <ListBox Name="lstbox"  Width="150"  Height="150"  Margin="24"  SelectionChanged="ListBoxOnSelection" />  <x:Code>  <![CDATA[  void WindowOnLoaded(object sender, RoutedEventArgs args)  {  foreach (System.Reflection.PropertyInfo prop in  typeof(Brushes).GetProperties())  lstbox.Items.Add(prop.Name);  }  void ButtonOnClick(object sender, RoutedEventArgs args)  {  Button btn = sender as Button;  MessageBox.Show("The button labeled '" +  btn.Content +  "' has been clicked.");  }  void ListBoxOnSelection(object sender, SelectionChangedEventArgs args)  {  string strItem = lstbox.SelectedItem as string;  System.Reflection.PropertyInfo prop =  typeof(Brushes).GetProperty(strItem);  elips.Fill = (Brush)prop.GetValue(null, null);  }  ]]>  </x:Code>  </StackPanel>  </Window> |

* 내장된 코드는 x:Code 엘리먼트와 x:Code 엘리먼트 안에 CDATA 구역이 필요하다. CDATA 구역은 항상 “<![CDATA[“ 문자열로 시작되며, “]]>” 문자열로 끝난다. 그렇기 때문에 어떠한 상황에서도 “]]>” 문자열이 있을 수 없으며, 혼동되는 코드에는 공백을 넣어서 해결 한다.
* XAML 파일에 내장코드는 특수한 상황에서는 편하지만, 이것을 일반적인 목적의 코딩에서 는 사용하지 말아야 한다.

11번 예제 624p

* Page로 빌드 작업을 가진 Window를 위한 XAML 파일로 XAML 파일을 이용한 UI 작성의 하나의 예이다.

|  |
| --- |
| <!-- ======================================================  DesignXamlButtonApp.xaml (c) 2006 by Charles Petzold  ====================================================== -->  <Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  StartupUri="DesignXamlButtonWindow.xaml" /> |

|  |
| --- |
| <!-- =========================================================  DesignXamlButtonWindow.xaml (c) 2006 by Charles Petzold  ========================================================= -->  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  Title="Design XAML Button"  SizeToContent="WidthAndHeight"  ResizeMode="CanMinimize">  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24">  <StackPanel>  <Polyline Stroke="Black"  Points="0 10,10 0,20 10,30 0,40 10,50 0,  60 10,70 0,80 10,90 0,100 10" />  <Image Margin="0,10,0,0"  Source="BOOK06.ICO"  Stretch="None" />  <Label HorizontalAlignment="Center">  \_Read Books!  </Label>  <Polyline Stroke="Black"  Points="0 0,10 10,20 0,30 10,40 0,50 10,  60 0,70 10,80 0,90 10,100 0" />  </StackPanel>  </Button>  </Window> |