20장 프로퍼티와 속성

- 많은 XAML 엘리먼트는 클래스와 구조체를 참조하고 그 결과로 객체를 생성하는데, 이것을 객체 엘리먼트(object elements, 비주얼 스튜디오에선 object 요소)라고 함.

- 엘리먼트는 이 객체상에서 프로퍼티를 설정하는 객체를 가지고 있는데, 이것을 프로퍼티 속성(property attributes)이라고 함.

- 자식 엘리먼트를 포함하는 대안적인 문법으로 프로퍼티를 명세할 수도 있는데, 이것은 프로퍼티 엘리먼트(property elements, 비주얼 스튜디오에선 속성 요소)라고 한다.

- 프로퍼티 엘리먼트는 엘리먼트 이름과 프로퍼티 이름 사이에 점으로 표현.

1번째 예제

|  |
| --- |
| <!-- =======================================  Star.xaml (c) 2006 by Charles Petzold  ======================================= -->  <Polygon xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Points="144 48, 200 222, 53 114, 235 114, 88 222"  Fill="Red"  Stroke="Blue"  StrokeThickness="5" /> |

Polygon은 Pointcollection 타입의 Points라는 프로퍼티를 포함하고 잇는데 PointcollectionConverter는 연속적인 X와 Y좌표를 포인트로 명세할수 있게 해준다

|  |
| --- |
| Fill="Red" 은 Fill = “#FF0000”인 16진법으로 RGB색상 값을 사용할 수도 잇고  Fill = “sc#0.5,1,0,0”인 부동소수점을 사용해 빨간색,녹색,파란색 값을 scRGB 스킴으로  표현 할수도 잇다 |

Fill프로 퍼티를 사용하지 않고 LinearGradientBrush를 사용해 XAML속에서 명세하는법

|  |
| --- |
| <Polygon xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Points="144 48, 200 222, 53 114, 235 114, 88 222"    Stroke="Blue"  StrokeThickness="5" >  <Polygon.Fill>  Red  </Polygon.Fill>  </Polygon> |

Polygon 태크 내의 Fill 속성을 제거하고 Polygon.Fill이란 이름의 자식 엘리 먼트로 교체하고

이 엘리먼트의 컨텐트를 문자열 Red로 설정한다.

많은 XAML 엘리먼트는 클래스와 구조체를 참조하고 그 결과로 객체를 생서하는데, 이것을

엘리먼트라고 한다.

(엘리먼트의 예)

|  |
| --- |
| <Polygon…/> = 객체 엘리먼트 |

엘리먼트는 이 객체상에서 프로퍼티를 설정하는 객체를 가지고 있는데,이것을 프로퍼티 속성

이라고 한다.

(프로퍼티 속성의 예)

|  |
| --- |
| Fill = “Red” |

또한 자식 엘리먼트를 포함하는 대안적인 문법으로 프로퍼티를 명세할 수도 있는데 이것을 프로퍼티 엘리먼트라고 한다

(프로퍼티 엘리먼트의 예)

|  |
| --- |
| <Polygon.Fill>  Red  </Polygon.Fill> |

또한 엘리먼트는 반드시 Brush로 변환될 수 있는 컨텐트를 가져야 한다

예)

|  |
| --- |
| <Polygon.Fill>  Red  </Polygon.Fill> |
| <Polygon.Fill>  #FF0000  </Polygon.Fill> |

좀더 길게 표현해 Polygon.Fill 컨텐트를 더 명확하게 표현하면

|  |
| --- |
| <Polygon.Fill>  <Brush>  Red  </Brush>  </Polygon.Fill> |

Polygon.Fill 프로퍼티 엘리먼트의 컨텐트는 이제”Red” 문자열의 Brush 객체 엘리먼트인데

이것은 SolidColorBrush 타입의 객체로 변환 될 수 있기 때문에 다음과 같이 표현이 가능하다.

|  |
| --- |
| <Polygon.Fill>  <Solidcolor Brush >  Red  </Solidcolor Brush >  </Polygon.Fill> |

SolidColorBrush에는 Color라는 이름의 프로퍼티가 있는데 ColorConverter는 Brush 컨버터와 같은 변환을 수행하기 때문에 SolidColorBrush의 Color 프로퍼티를 프로퍼티 속성으로 설정할수 있다

|  |
| --- |
| <Polygon xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Points="144 48, 200 222, 53 114, 235 114, 88 222"  Stroke="Blue"  StrokeThickness="5" >  <Polygon.Fill>  <SolidColorBrush Color= "Red"/>  </Polygon.Fill>  </Polygon> |

이것을 프로퍼티 엘리먼트로 확장하면

|  |
| --- |
| <Polygon xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Points="144 48, 200 222, 53 114, 235 114, 88 222"  Stroke="Blue"  StrokeThickness="5" >  <Polygon.Fill>  <SolidColorBrush>  <SolidColorBrush.Color>  Red  </SolidColorBrush.Color>  </SolidColorBrush>  </Polygon.Fill>  </Polygon> |

색상은 byte 타입의 A,R,G,B라는 프로퍼티를 갖는데 이 프로퍼티를 Color 태그 내에서는 10진수나 16진수의 문법으로 다음과 같이 설정할수 있다

|  |
| --- |
| <Polygon xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Points="144 48, 200 222, 53 114, 235 114, 88 222"  Stroke="Blue"  StrokeThickness="5" >  <Polygon.Fill>  <SolidColorBrush>  <SolidColorBrush.Color>  <Color A="255" R="#FF" G="0" B="0">  </Color>  </SolidColorBrush.Color>  </SolidColorBrush>  </Polygon.Fill>  </Polygon> |

여기까지 설명한 문법이 Fill 프로퍼티에 그라이언트 브러시를 정의한 것

LinearGradientBrush는 StartPoint와 EndPoint라는 두개의 프로퍼티를 가지고 있는데

|  |
| --- |
| <Polygon xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Points="144 48, 200 222, 53 114, 235 114, 88 222"  Stroke="Blue"  StrokeThickness="5" >  <Polygon.Fill>  <LinearGradientBrush StartPoint="0 0" EndPoint="1 0">  <GradientStop Offset="0" Color="Red"/>  <GradientStop Offset="0.5" Color="Green"/>  <GradientStop Offset="1" Color="Blue"/>  </LinearGradientBrush>  </Polygon.Fill>  </Polygon> |

Polygon.Fill 프로퍼티는 LinearGradientBrush 타입의 객체로서 StartPoint, EndPoint, GradientStop라는

프로퍼티와 3개의 GradientStop 객체 컬렉션을 포함하면 그 GradientStop에는 Offset과 Color라는 프로퍼티가 있다

2번예제

|  |
| --- |
| <Polygon xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Points="144 48, 200 222, 53 114, 235 114, 88 222"  Stroke="Blue"  StrokeThickness="5">  <Polygon.Fill>  <RadialGradientBrush>  <GradientStop Offset="0" Color="Blue" />  <GradientStop Offset="1" Color="Red" />  </RadialGradientBrush>  </Polygon.Fill>  </Polygon> |

3번예제

|  |
| --- |
| //--------------------------------------------------------------  // DumpContentPropertyAttributes.cs (c) 2006 by Charles Petzold  //--------------------------------------------------------------  using System;  using System.Collections.Generic;  using System.Reflection;  using System.Windows;  using System.Windows.Markup;  public class DumpContentPropertyAttributes  {  [STAThread]  public static void Main()  {  // PresentationCore 와 PresentationFramework 가 로드 되는지 확인  UIElement dummy1 = new UIElement();  FrameworkElement dummy2 = new FrameworkElement();  // 컨텍트 프로퍼티와 클래스를 저장할 SortedList  SortedList<string, string> listClass = new SortedList<string, string>();  // 문자열 포멧팅  string strFormat = "{0,-35}{1}";  // 로드한 어셈블리를 순회하는 루프  foreach (AssemblyName asmblyname in  Assembly.GetExecutingAssembly().GetReferencedAssemblies())  {  // 타입루프  foreach (Type type in Assembly.Load(asmblyname).GetTypes())  {  // 커스텀속성루프  // (상속받지 않은 것에 한해 false로 인자를 설정!)  foreach (object obj in type.GetCustomAttributes(  typeof(ContentPropertyAttribute), true))  {  // ContentPropertyAttribute면 리스트에 추가!  if (type.IsPublic && obj as ContentPropertyAttribute != null)  listClass.Add(type.Name,  (obj as ContentPropertyAttribute).Name);  }  }  }  // 결과출력  Console.WriteLine(strFormat, "Class", "Content Property");  Console.WriteLine(strFormat, "-----", "----------------");  foreach (string strClass in listClass.Keys)  Console.WriteLine(strFormat, strClass, listClass[strClass]);  }  } |

이 예제는 XAML을 작성할 때 프로퍼티가 얼마나 중요한지 보았는데 모튼 클래스의 ContentProperty 속성과 그 컨텐트 프로퍼티의 정보를 제공하는 프로그램이다.

4번예제)

이것은 6장의 DockAroundTheBlock 프로그램이다

|  |
| --- |
| //---------------------------------------------------  // DockAroundTheBlock.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.DockAroundTheBlock  {  class DockAroundTheBlock : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new DockAroundTheBlock());  }  public DockAroundTheBlock()  {  Title = "Dock Around the Block";  // 1. 패¡¨¢널øI 생íy성ù¨¬ 및ö¡¿ 초E기¾a화¡©  DockPanel dock = new DockPanel();  Content = dock;  // 2. 원¯©ª하I는¥A 자U식öA 객Æ¢¥체¨ù(이I미öI지o, 컨A트¡¢ç롤¤N, 패¡¨¢널øI..)  for (int i = 0; i < 17; i++)  {  Button btn = new Button();  btn.Content = "Button No. " + (i + 1);  //-------------------------------------------------  dock.Children.Add(btn);    btn.SetValue(DockPanel.DockProperty, (Dock)(i % 4));    DockPanel.SetDock(btn, (Dock)(i % 4));  //-------------------------------------------------    }    // dock.LastChildFill = true;  dock.LastChildFill = false;  }  }  } |

이것을 비슷한 자멜로 바구면

|  |
| --- |
| <!-- =========================================================  AttachedPropertiesDemo.xaml (c) 2006 by Charles Petzold  ========================================================= -->  <DockPanel xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">  <Button Content="Button No. 1" DockPanel.Dock="Left" />  <Button Content="Button No. 2" DockPanel.Dock="Top" />  <Button Content="Button No. 3" DockPanel.Dock="Right" />  <Button Content="Button No. 4" DockPanel.Dock="Bottom" />  <Button Content="Button No. 5" />  </DockPanel> |

Grid는 특히 XAML에서 잘 사용되는데 그것은 열과 행을 정의하는 RowDefinition과

ColumnDefinition이 각기 Grid.RowDefinitions 와 Grid.ColumnDefinitions 프로퍼티 속에서 한줄로 표현이 가능해 C#처럼 장황하지 않다

|  |
| --- |
| <!-- =============================================  SimpleGrid.xaml (c) 2006 by Charles Petzold  ============================================= -->  <Grid xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">  <Grid.RowDefinitions>  <RowDefinition Height="100" />  <RowDefinition Height="Auto" />  <RowDefinition Height="100\*" />  </Grid.RowDefinitions>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="33\*" />  <ColumnDefinition Width="Auto" />  <ColumnDefinition Width="67\*" />  </Grid.ColumnDefinitions>  <Button Content="Button No. 1" Grid.Row="0" Grid.Column="0" />  <Button Content="Button No. 2" Grid.Row="1" Grid.Column="0" />  <Button Content="Button No. 3" Grid.Row="2" Grid.Column="0" />  <GridSplitter HorizontalAlignment="Center" Width="6"  Grid.Row="0" Grid.Column="1" Grid.RowSpan="3" />  <Button Content="Button No. 4" Grid.Row="0" Grid.Column="2" />  <Button Content="Button No. 5" Grid.Row="1" Grid.Column="2" />  <Button Content="Button No. 6" Grid.Row="2" Grid.Column="2" />  </Grid> |

어떤경우에는 프로퍼티가 정의되어 있지 않은 엘리먼트 속에서 클래스 이름과 프로퍼티 이름을 포함하는 속성을 사용할수 있다

5번예제)

|  |
| --- |
| <StackPanel xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  HorizontalAlignment="Center"  TextBlock.FontSize="16pt"  TextBlock.Foreground="Blue" >  <TextBlock>  Just a TextBlock  </TextBlock>  <Button>  Just a Button  </Button>  </StackPanel> |

이 XAML 파일은 StackPanel 엘리먼트 안에서 FontSize 와 Foreground 프로퍼티를 설정하는데 StackPanel이 이 프로퍼티를 가지지 않기 때문에 이프로퍼티가 정의된 클래스가 프로퍼티 앞에 명시되어야 한다

6번예제)

클래스와 이벤트 이름을 속성으로 사용해서 이벤트 핸들러를 설정할수도 있는데 이랜들러는 자식 엘리먼트에게 영향을 미치게 된다

|  |
| --- |
| //--------------------------------------------------  // RoutedEventDemo.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.RoutedEventDemo  {  public partial class RoutedEventDemo : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new RoutedEventDemo());  }  public RoutedEventDemo()  {  InitializeComponent();  }  void MenuItemOnClick(object sender, RoutedEventArgs args)  {  string str = (args.Source as MenuItem).Header as string;  Color clr = (Color)ColorConverter.ConvertFromString(str);  txtblk.Foreground = new SolidColorBrush(clr);  }  }  } |

RoutedEventDemo.xaml

|  |
| --- |
| <!-- ==================================================  RoutedEventDemo.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.RoutedEventDemo.RoutedEventDemo"  Title="Routed Event Demo">  <TextBlock Name="txtblk"  FontSize="24pt"  HorizontalAlignment="Center"  VerticalAlignment="Center"  ToolTip="Right click to display context menu">  TextBlock with Context Menu  <TextBlock.ContextMenu>  <ContextMenu MenuItem.Click="MenuItemOnClick">  <MenuItem Header="Red" />  <MenuItem Header="Orange" />  <MenuItem Header="Yellow" />  <MenuItem Header="Green" />  <MenuItem Header="Blue" />  <MenuItem Header="Indigo" />  <MenuItem Header="Violet" />  </ContextMenu>  </TextBlock.ContextMenu>  </TextBlock>  </Window> |

다음 RoutedEventDemo 프로젝트의 경우 2개의 파일을 포함하는데, RoutedEventDemo.cs 와 RoutedEventDemo.xaml이다. 다음 XAML파일은 ContextMenu 엘리먼트안에 MenuItem.Click 속성을 포함하고 있고, 핸들러는 TextBlock의 컨텍스트 메뉴를 가지는 MenuItem 엘리먼트에 적용된다.