**제 8장. 의존 프로퍼티(Dependency Property)**

1. 의존 프로퍼티란(Dependency Property)?  
: 한글로 번역하게 되면 의존 속성으로 번역이 가능한 용어이다. 이름 그대로 무언가에 의존하는 속성이라는 뜻이다. 즉, **1-1).**다른 입력 값을 기준으로 하여 속성 값을 계산하기 위한 용도로 사용하며 애니메이션, 스타일링, 데이터 바인딩 등 많은 부분에서 사용되어 지고 있다.  
이러한 의존 프로퍼티에는 3가지의 구성요소가 존재한다.  
**1. DependencyObject : ‘**의존 속성'을 가지는 객체이다.  
WPF의 대부분의 UI 요소들은 이 Dependency Object를 상속받아 구현되어 있다. 이 Dependency Object는 GetValue 와 SetValue 메소드가 있는데, 모든 '의존 속성'은 **1-2).GetValue 와 SetValue를 통해 구현**되어 있다.  
  
**2. DependencyProperty :** '의존 속성' 자체를 정의한 것이다.  
예를 들어 Button의 Background Brush에 대한 Dependency Property가 있고, 또한 Brush의 Color에 대한 Dependency Property도 있다. (Brush도 DependencyObject 이다.)  
  
**3. PropertyMetadata :** Dependency Property의 부가 정보 정도로 생각하면 된다.

**여기서 잠깐 1-1.**

다른 입력 값 이란  Theme, 기본설정(속성), DataBinding, Animation, 리소스 및 스타일,   
부모 요소에서 상속받은 속성 값 등을 말함.)

**여기서 잠깐1-2.**

**GetValue와 SetValue는 어디에서?**

의존 프로퍼티의 3가지 구성요소 중 Dependency Object에서 GetValue와 SetValue를 통하여 구현된다고 말하였는데 **DependencyProperty는 DependecyObject의 static 클래스 단위로** **정의**되어 있기 때문에 Get Value와 SetValue가 사용이 가능한 것이다. 뿐만 아니라 **WPF의 많은 클래스들을 자손(자식)으로 두고 있기에** GetValue와 SetValue를 사용할 수 있는 것이다.

|  |
| --- |
| Object    ㄴDispatcherObject (abstract)       ㄴ**DependencyObject**           ㄴVisual (abstract)              ㄴUIElement                 ㄴFrameworkElement                    ㄴControl |

2. 의존 프로퍼티 선언  
의존 프로퍼티는 .NET CLR과 사용법에 있어서 차이점이 있는데 그것은 바로 사용하기 하기 전에 반드시 **DependencyProperty.Register**를 통해서 실버라이트 런타임에게 이 속성의 의존 속성으로써 사용한다는 것을 알려줘야 한다는 것이다.  
간단한 예제소스를 통하여 .NET CLR과 의존 프로퍼티의 차이점을 짚어보고 넘어가겠다.

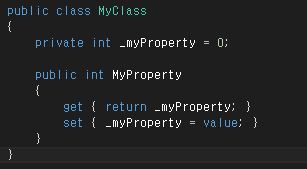
  
그림2-1) .NET CLR의 get, set 사용법

  
그림2-2) 의존 프로퍼티의 get, set 사용법

그림 2-1과 2-2를 보았을 때 느껴지는 **2-1).**차이점이 있다면 의존 프로퍼티가 어떻게 사용되어지는 이해를 제대로 했다는 것이다. 바로 .NET CLR 속성은 속성값을 자기 자신 안에 저장해 둔다는 것이며, 의존 프로퍼티는 **2-2).DependencyProperty.Register에 속성값을 등록하여  
PropertMetadata클래스를 이용하여 정의**한다는 것이다.

**여기서 잠깐2-1.  
의존 프로퍼티와 .NET CLR의 차이점 3가지**1) 의존 속성에서는 속성값을 **GetValue**와 **SetValue**를 통해서 읽거나 쓴다.

2) **DependencyProperty.Register**를 이용하여 속성을 등록한다. 이 때 속성의 타입과 클래스 타입을 파라미터로 지정한다.

3) 의존 속성의 기본값은 **PropertyMetaData**클래스를 이용하여 정의한다.

해당 차이점을 그림으로 표현하면 다음과 같다.

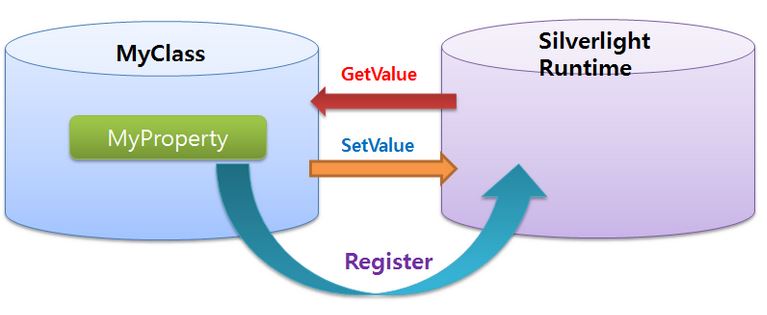


그림 2-3) 의존 프로퍼티 속성 값 정의 및 사용

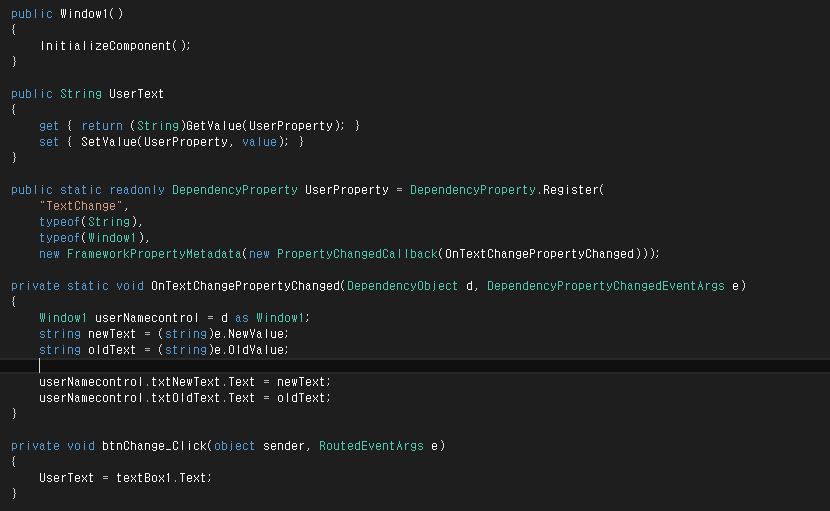
**여기서 잠깐2-2.  
DependencyProperty**.**Register**함수에 선언하여야 하는 4개의 파라미터

|  |  |
| --- | --- |
| 속성 | 설명 |
| Name | 등록할 속성의 이름 |
| PropertyType | 속성의 타입 |
| DependencyObject(ownerType) | 속성을 포함한 부모의 타입 |
| PropertyMetadata(typeMetadata) | 속성 메타데이터 지정  (보통 콜백(Callback) 함수를 지정함) |

예제소스)

Dependencyproperty\_Test로 이름을 주어 생성하였다.

**MainWindow.xaml.cs**



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Dependencyproperty\_Test

{

    public partial class Window1 : Window

    {

        public Window1()

        {

            InitializeComponent();

        }

        public String UserText

        {

            get { return (String)GetValue(UserProperty); }

            set { SetValue(UserProperty, value); }

        }

        public static readonly DependencyProperty UserProperty = DependencyProperty.Register(

            "TextChange",

            typeof(String),

            typeof(Window1),

            new FrameworkPropertyMetadata(new PropertyChangedCallback(OnTextChangePropertyChanged)));

//의존프로퍼티 사용

        private static void OnTextChangePropertyChanged(DependencyObject d, DependencyPropertyChangedEventArgs e)

        {

            Window1 userNamecontrol = d as Window1;

            string newText = (string)e.NewValue;

            string oldText = (string)e.OldValue;

            userNamecontrol.txtNewText.Text = newText;

            userNamecontrol.txtOldText.Text = oldText;

        }

        private void btnChange\_Click(object sender, RoutedEventArgs e)

        {

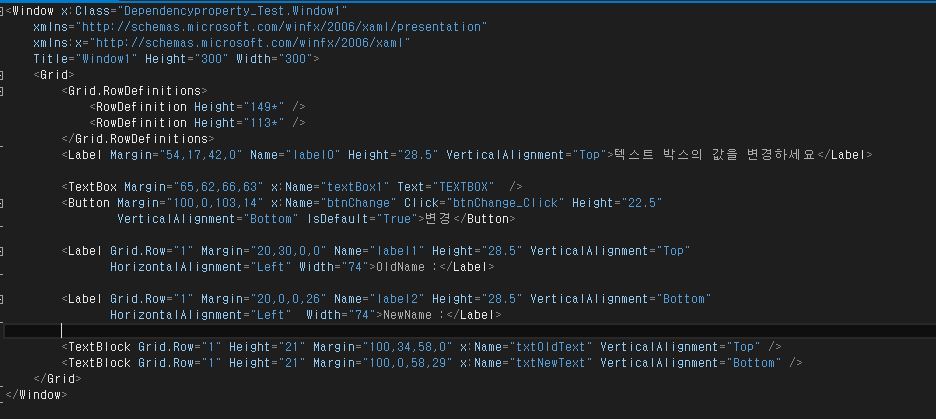
            UserText = textBox1.Text;

        }

    }

}

**MainWindow.xaml**



<Window x:Class="Dependencyproperty\_Test.Window1"

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

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

    Title="Window1" Height="300" Width="300">

    <Grid>

        <Grid.RowDefinitions>

            <RowDefinition Height="149\*" />

            <RowDefinition Height="113\*" />

        </Grid.RowDefinitions>

        <Label Margin="54,17,42,0" Name="label0" Height="28.5" VerticalAlignment="Top">텍스트 박스의 값을 변경하세요</Label>

        <TextBox Margin="65,62,66,63" x:Name="textBox1" Text="TEXTBOX"  />

        <Button Margin="100,0,103,14" x:Name="btnChange" Click="btnChange\_Click" Height="22.5" VerticalAlignment="Bottom" IsDefault="True">변경</Button>

        <Label Grid.Row="1" Margin="20,30,0,0" Name="label1" Height="28.5" VerticalAlignment="Top" HorizontalAlignment="Left" Width="74">OldName :</Label>

        <Label Grid.Row="1" Margin="20,0,0,26" Name="label2" Height="28.5" VerticalAlignment="Bottom" HorizontalAlignment="Left"  Width="74">NewName :</Label>

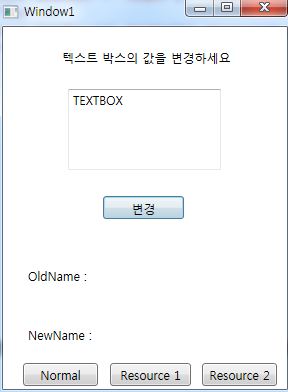
        <TextBlock Grid.Row="1" Height="21" Margin="100,34,58,0" x:Name="txtOldText" VerticalAlignment="Top" />

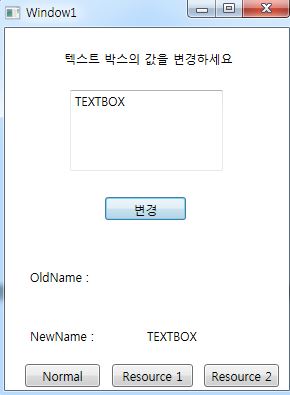
        <TextBlock Grid.Row="1" Height="21" Margin="100,0,58,29" x:Name="txtNewText" VerticalAlignment="Bottom" />

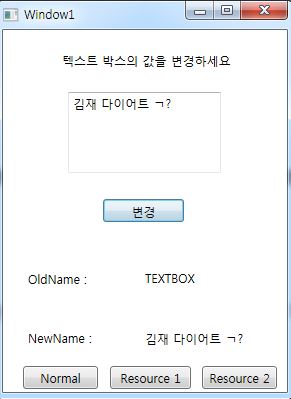
    </Grid>

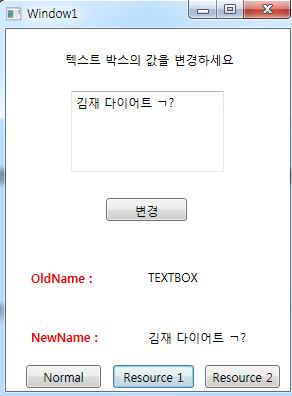
</Window>

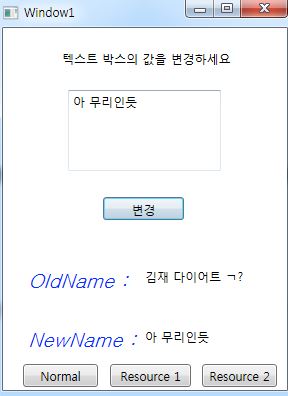
와 같이 소스를 구성할 시 다음과 같이 의존프로퍼티를 사용한 oldText와 newText가 변경되는 것을 알 수 있을 것이다.

  
예제소스 구현 그림1.

  
예제소스 구현 그림2.

  
예제소스 구현 그림3.

  
번외 구현 그림1.

  
번외 구현 그림2.