리소스

# 이전의 리소스

프로젝트 내의 특정 파일에 빌드 작업을 리소스로 설정해 컴파일

하는 방법. 대부분 아이콘이나 비트맵과 같은 바이너리 파일이 일반적.

이 어셈블리 리소스는 어셈블리(실행 파일이나 동적 링크 라이브

러리)에 저장하거나 실제 리소스 파일 이름을 참조하는 Uri객체를

정의하는 것으로 접근할 수 있다.

# 현재의 리소스

때로는 지역적으로 정의된 형태의 리소스로 참조되는데 , 그 이유는 그것이 XAML 내에서 정의되고(때로는 C# 코드 내에서) , 일반적으로 애플리케이션 내의 엘리먼트 , 컨트롤 , 페이지 , 창과 관련이 있기 때문이다. 이 리소스는 오직 리소스가 정의된 엘리먼트 내에서나 그 엘리먼트 자식 내에서만 사용이 가능하며 , 이것이 XAML에서 C# 정적 읽기전용 필드를 대체하는 대안쯤으로 생각 할 수 있을 것이다.   
정적 읽기 전용 필드와 같이 리소스는 실행 시에 한 번 생성되며 , 그것을 참조하는 엘리먼트에 공유된다.

XAML은 리소스 키를 정의하기 위한 목적으로 x:Key 속성을 정의.

**1) 예제** StackPanel의 Resources 컬렉션 속에 있는 두 개의 폰트 크기 리소스를 어떻게

정의 하는지와 StackPanel의 자식 엘리먼트가 그 리소스를 어떻게 참조하는지

보여주는 예제.

|  |
| --- |
| FontSizeResources.xaml  <StackPanel x:Class="WpfApplication3.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:s="clr-namespace:System;assembly=mscorlib">  // XML 네이스페이스를 접두어로 s로 정의하는데 이것으로 리소스 컬렉션 내에서  // Double 구조체를 참조할 수 있다.  <StackPanel.Resources>  <s:Double x:Key="fontsizeLarge">  18.7  </s:Double>  <s:Double x:Key="fontsizeSmall">  14.7  </s:Double>  </StackPanel.Resources>  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24">  // 첫 번째 사용 방법 FontSize 프로퍼티 엘리먼트를 사용해 리소스를 참조하는데  // StaticResource 엘리먼트의 ResourcKey 속성에 원하는 항목의 키를 설정  <Button.FontSize>  <StaticResource ResourceKey="fontsizeLarge" />  </Button.FontSize>  Button with large FontSize  </Button>  // 두 번째 사용방법 FontSize 속성에 StaticResource와 키 이름을 명시한후  // 중괄호로 묶는 방법  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24"  FontSize="{StaticResource fontsizeSmall}" >  Button with small FontSize  </Button>  </StackPanel> |

**2) 예제** 한 리소스 컬렉션에서 모든 키는 서로 유일해야 하지만 2개의 서로 다른 리소스 컬렉션에서는 동일한 키가 사용될 수 있다. 리소스를 찾을 때 검색은 리소스를 참조하는 엘리먼트의 리소스 컬렉션에서 시작되어 키를 찾을 때까지 트리를 순회해 올라가는 과정을 설명한 예제.

|  |
| --- |
| ResourceLookDemo  <StackPanel x:Class="WpfApplication3.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Orientation="Horizontal">  <StackPanel.Resources>  <SolidColorBrush x:Key="brushText" Color="Blue" />  </StackPanel.Resources>  <StackPanel>  <StackPanel.Resources>  <SolidColorBrush x:Key="brushText" Color="Red" />  </StackPanel.Resources>  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24"  Foreground="{StaticResource brushText}">  Button with Red text  </Button>  </StackPanel>  <StackPanel>  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24"  Foreground="{StaticResource brushText}">  Button with Blue text  </Button>  </StackPanel>  </StackPanel> |

**3) 예제**프로퍼티의 문자열을 보여주기 위한 것이기 때문에 대부분의 프로퍼티는 문자열을 반환한다. Content 프로퍼티는 어떤 객체도 될 수 있고 설정하는 객체는 ToString 메소드로 표현.

|  |
| --- |
| EnvironmentInfo  <StackPanel x:Class="WpfApplication3.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:s="clr-namespace:System;assembly=mscorlib">  <TextBlock>  <Label Content="Operating System Version: " />  <Label Content="{x:Static s:Environment.OSVersion}" />  <LineBreak />  <Label Content=".NET Version: " />  <Label Content="{x:Static s:Environment.Version}" />  <LineBreak />  <Label Content="Machine Name: " />  <Label Content="{x:Static s:Environment.MachineName}" />  <LineBreak />  <Label Content="User Name: " />  <Label Content="{x:Static s:Environment.UserName}" />  <LineBreak />  <Label Content="User Domain Name: " />  <Label Content="{x:Static s:Environment.UserDomainName}" />  <LineBreak />  <Label Content="System Directory: " />  <Label Content="{x:Static s:Environment.SystemDirectory}" />  <LineBreak />  <Label Content="Current Directory: " />  <Label Content="{x:Static s:Environment.CurrentDirectory}" />  <LineBreak />  <Label Content="Command Line: " />  <Label Content="{x:Static s:Environment.CommandLine}" />  </TextBlock>  </StackPanel> |

**4) 예제**정적 필드나 프로퍼티를 프로젝트의 C# 코드 속에 정의하고 그것을 XAML 파일에서 접근하는 방법.

|  |
| --- |
| Constants.cs  public static class Constants  {  public static readonly FontFamily fntfam =  new FontFamily("Times New Roman Italic");  public static double FontSize  {  get { return 72 / 0.75; }  }  public static readonly LinearGradientBrush brush =  new LinearGradientBrush(Colors.LightGray, Colors.DarkGray,  new Point(0, 0), new Point(1, 1));  }  // 2개의 정적 읽기 전용 필드와 하나의 정적 읽기 전용 프로퍼티로 정의되어있는데 이것은  다양성을 위해서이기 때문에 특별한 의미는 없다.  WPF 어셈블리가 아닌 프로젝트의 소스 코드 부분에서 정의됐기 때문에 XAML파일  을 필드와 프로퍼티가 있는 클래스의 네임스페이스과 연결된 접두어를 XML 네임스페이스 선언에서 정의할 필요가 있다  AccessStaticFields.cs  public partial class AccessStaticFields : Window  {  [STAThread]  public static void Main()  {  Application app = new Application();  app.Run(new AccessStaticFields());  }  public AccessStaticFields()  {  InitializeComponent();  }  }  ////////////////////////////////////////////////////////////////////////////////////////////////////////////  AccessStaticFields.xaml  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:src="clr-namespace:example4"  //namespace를 참고하여<Constants.cs>를 포함한다.  x:Class="hyun.AccessStaticFields"  Title="Access Static Fields"  SizeToContent="WidthAndHeight">  <TextBlock Background="{x:Static src:Constants.brush}"  FontSize="{x:Static src:Constants.FontSize}"  TextAlignment="Center">  <TextBlock.FontFamily>  <x:Static Member="src:Constants.fntfam" />  </TextBlock.FontFamily>  Properties from<LineBreak />Static Fields  </TextBlock>  </Window> |

StaticResource : 키가 객체에 접근할 때 한 번 사용되고 접근한 객체를 유지.

DynamicResource: 키가 계속 유지되어 객체가 필요할 때마다 접근.

**5) 예제**

2개의 브러쉬를 정의하고 활성화된 제목 표시줄과 색상과 비활성화된 제목 표시풀 색상으로 표현한 그라디언트를 정의. 두번째 SolidColorBrush는 DynamicResource를 통해 SystemColors.ActiveCaptionColorKey를 사용

|  |
| --- |
| DynamicResourceDemo  <StackPanel x:Class="WpfApplication1.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Background="{DynamicResource  {x:Static SystemColors.InactiveCaptionBrushKey}}">  <StackPanel.Resources>  <LinearGradientBrush x:Key="dynabrush1"  StartPoint="0 0" EndPoint="11">  <LinearGradientBrush.GradientStops>  <GradientStop Offset="0"  Color="{DynamicResource  {x:Static SystemColors.ActiveCaptionColorKey}}" />  //2개의리소스 SystemColors.ActiveCaptionColorKey와 SystemColors.InactiveCaptionColorKey  를 참조하기 위해 DynamicResource를 사용  이것은 참조를 위한 키지만 브러시를 참조하지 않고 색을 참조했다. 왜냐하면 이것이 두  GradientStop 객체의 Color 프로퍼티를 설정해야 하기 때문이다.  <GradientStop Offset="1"  Color="{DynamicResource  {x:Static SystemColors.InactiveCaptionColorKey}}" />  </LinearGradientBrush.GradientStops>  </LinearGradientBrush>  <SolidColorBrush x:Key="dynabrush2"  Color="{DynamicResource  {x:Static SystemColors.ActiveCaptionColorKey}}" />  </StackPanel.Resources>  <Label HorizontalAlignment="Center"  FontSize="96"  Content="Dynamic Resources"  Background="{StaticResource dynabrush1}"  Foreground="{StaticResource dynabrush2}" />  </StackPanel>  // 라벨의 Background와 Foreground 속성을 DynamicResource로 변경하면 프로그램은 더 이상시스템 색의 변경에 반응하지 않는다. 문제는 DynamicResource는 키에 의해 참조된 객체가 재생성되기를 바라는데, 실제 참조한 브러시 객체는 재생성 되지 않는다.  따라서 DynamicResource는 Background와 foreground 프로퍼티를 갱신하는데 전혀 영향을 못미친다. |

**6) 예제**시스템 색이나 시스템 설정과 연관된 키를 자신만의 리소스 정의에서 사용할 수도 있는 이 경우 지역 리소스 정의가 시스템 설정에 우선한다.

|  |
| --- |
| AnotherResourceLookupDemo  <StackPanel x:Class="WpfApplication1.MainWindow"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  Orientation="Horizontal">  <StackPanel>  <StackPanel.Resources>  <SolidColorBrush  x:Key="{x:Static SystemColors.ActiveCaptionBrushKey}"  Color="Red" />  </StackPanel.Resources>  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24"  Foreground="{DynamicResource  {x:Static SystemColors.ActiveCaptionBrushKey}}">  Button with Red text  </Button>  </StackPanel>  <StackPanel>  <Button HorizontalAlignment="Center"  VerticalAlignment="Center"  Margin="24"  Foreground="{DynamicResource  {x:Static SystemColors.ActiveCaptionBrushKey}}">  Button with Blue text  </Button>  </StackPanel>  </StackPanel>  // StackPanel 속에 있는 버튼은 빨간색 브러시를 가지므로 버튼의 문자열이 빨간색으로 표현되고 나머지 버튼은 활성화된 제목 표시줄 색의 브러시를 가지기 때문에 시스템 색을 변경할 때처럼 버튼의 색도 동적으로 변경된다. 따라서 스탠드얼론 리소스 정의가 시스템 설정에 우선함을 알 수 있다. |

**7) 예제**다수의 프로젝트에서 공유하고 싶은 리소스는 ResourceDictionary를 루트 엘리먼트로 하는 XAML 파일에 모을 수 잇는데 , 리소스는 이 루트 엘리먼트의 자식이 된다. 다음은 하나의 리소스를 포함하는 리소스 사전 예제이다.

|  |
| --- |
| MyResources1  <ResourceDictionary  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">  <LinearGradientBrush x:Key="brushLinear">  <LinearGradientBrush.GradientStops>  <GradientStop Color="Pink" Offset="0" />  <GradientStop Color="Aqua" Offset="1" />  </LinearGradientBrush.GradientStops>  </LinearGradientBrush>  </ResourceDictionary>  ////////////////////////////////////////////////////////////////////////////////////////////////////////////  MyResources2  <ResourceDictionary  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml">  <RadialGradientBrush x:Key="brushRadial">  <RadialGradientBrush.GradientStops>  <GradientStop Color="Pink" Offset="0" />  <GradientStop Color="Aqua" Offset="1" />  </RadialGradientBrush.GradientStops>  </RadialGradientBrush>  </ResourceDictionary>  ////////////////////////////////////////////////////////////////////////////////////////////////////////////  UseCommonResourcesApp  <Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  StartupUri="UseCommonResourcesWindow.xaml">  <Application.Resources>  <ResourceDictionary>  <ResourceDictionary.MergedDictionaries>  <ResourceDictionary Source="MyResources1.xaml" />  <ResourceDictionary Source="MyResources2.xaml" />  </ResourceDictionary.MergedDictionaries>  </ResourceDictionary>  </Application.Resources>  </Application>  UseCommonResouresWindow  <Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  Title="Use Common Resources"  Background="{StaticResource brushLinear}">  // 첫 번째 MyResources1의 색상타입  <Button FontSize="96pt"  HorizontalAlignment="Center"  VerticalAlignment="Center"  Background="{StaticResource brushRadial}">  // 두 번째 MyResources2의 색상타입  Button  </Button>  </Window> |