XAML Do-It-Yourself シリーズ

第 5 回 リソース

Microsoft®

XAML Do-It-Yourself
第 5 回 リソース

XAML Do-It-Yourself 第 5 回は、XAML を記述するうえで欠かせない、リソースについて学んでいきます。今回は、リソースについて次のことを学習します。

・リソースの記述方法
・リソースの有効範囲

■リソースとは

XAML では、アプリケーションで使用するデータ、ブラシなどの描画オブジェクト、外見を記述したスタイルなどをリソースとして扱います (スタイルについては次回で学習します)。XAML でリソースを記述する利点は、リソースとして定義したオブジェクトを XAML ファイル内で共有できる点です。

WPF で利用可能なほとんどのコントロールは、コントロール内で使用するリソースを保持するための Resources プロパティを持っています。Resources プロパティには複数のリソースを格納できます。

■ブラシによる塗りつぶし

例えばアプリケーションでグラデーションを使ったブラシを使用したいとします。通常、コントロールの背景を塗りつぶしたり、外枠 (Border) の部分を描画したりするにはブラシ (Brush) を使用します。

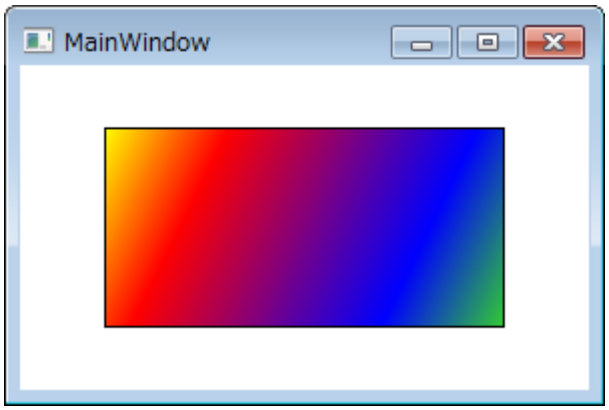
グラデーション効果のあるブラシは、次のようにして定義します。LinearGradientBrush は、線形グラデーション描画を行うブラシで、以下の記述では、描画領域の左上から右下にかけて、Yellow、Red、Blue、LimeGreen の色でグラデーションを描画します。

```
<LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
  <GradientStop Color="Yellow" Offset="0.0" />
  <GradientStop Color="Red" Offset="0.25" />
  <GradientStop Color="Blue" Offset="0.75" />
  <GradientStop Color="LimeGreen" Offset="1.0" />
</LinearGradientBrush>
```

リソースを用いずに、これを Rectangle (矩形) の内側の塗りつぶしに利用する場合は、次のように記述します。この場合には Fill プロパティにブラシを設定しますから、XAML の記述では、<Rectangle.Fill> 要素の中に記述することになります。

```
<Window x:Class="WpfApplication5.MainWindow"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Title="MainWindow" Height="200" Width="300">
  <Grid>
    <Rectangle Width="200" Height="100" Stroke="Black">
      <Rectangle.Fill>
        <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
```

```
        <GradientStop Color="Yellow" Offset="0.0" />
        <GradientStop Color="Red" Offset="0.25" />
        <GradientStop Color="Blue" Offset="0.75" />
        <GradientStop Color="LimeGreen" Offset="1.0" />
      </LinearGradientBrush>
    </Rectangle.Fill>
  </Rectangle>
 </Grid>
</Window>
```



■リソースの記述

これに対して、ウィンドウに複数のコントロールを配置し、それぞれのコントロールで同じブラシを使って描画するような場合は、ブラシをリソースとして定義し、それを使い回すことができます。

上で説明した矩形を表示するアプリケーションで、ブラシをリソースとして記述し、それにより矩形を塗りつぶすには次のように記述します。

```
<Window x:Class="WpfApplication5.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MainWindow" Height="200" Width="300">
  <Window.Resources>
    <LinearGradientBrush x:Key="myBrush" StartPoint="0,0" EndPoint="1,1">
      <GradientStop Color="Yellow" Offset="0.0" />
      <GradientStop Color="Red" Offset="0.25" />
      <GradientStop Color="Blue" Offset="0.75" />
      <GradientStop Color="LimeGreen" Offset="1.0" />
    </LinearGradientBrush>
  </Window.Resources>
  <Grid>
    <Rectangle Width="200" Height="100" Stroke="Black" Fill="{StaticResource
myBrush}" />
  </Grid>
</Window>
```

実行結果は前述の XAML の場合と変わりませんが、ここでは <Window> 要素の Resources プロパテ

ィにブラシ (LinearGradientBrush) を記述しています。

コントロールに直接ブラシを記述する場合と異なるのは、x:Key 属性を使ってこのブラシの定義に名前を付けている点です (x:Key="myBrush")。これによって <Window> 要素の下位にあるコントロールは、プロパティに "myBrush" を指定することで、このブラシが利用できるようになります。

そして <Rectangle> 要素では、myBrush を利用することを示すために、Fill プロパティに、

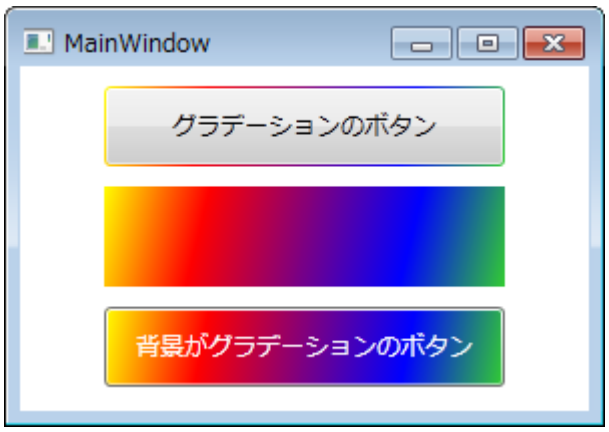
Fill="{StaticResource myBrush}"

という記述を行います。これは、領域の塗りつぶしに "myBrush" を利用することを示しています。

"StaticResource" は上記のリストで示すような、XAML ファイル内で定義したリソースを指定する場合に使うものと考えてよいでしょう。それに対して "DynamicResource" としてリソースを参照する場合があります。これは、アプリケーションのコード ビハインド部分のロジックで実行時に追加したリソースや、システム フォント、システム カラーなど実行時のシステムに関連するリソースを指定する場合に使います。

続いて、リソースとして記述したグラデーション ブラシを使って、ウィンドウに配置した 2 つのボタンと矩形の外見を変更してみましょう。最初の Button は外枠に、2 番目の Rectangle は内部の塗りつぶしに、3 番目の Button はボタンの背景に、このブラシを使用しています。

```
<Window x:Class="WpfApplication5.MainWindow"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Title="MainWindow" Height="210" Width="300">
  <Window.Resources>
    <LinearGradientBrush x:Key="myBrush" StartPoint="0,0" EndPoint="1,1">
      <GradientStop Color="Yellow" Offset="0.0" />
      <GradientStop Color="Red" Offset="0.25" />
      <GradientStop Color="Blue" Offset="0.75" />
      <GradientStop Color="LimeGreen" Offset="1.0" />
    </LinearGradientBrush>
  </Window.Resources>
  <StackPanel >
    <Button Margin="10" DockPanel.Dock="Top" Width="200" Height="40"
BorderBrush="{StaticResource myBrush}">グラデーションのボタン</Button>
    <Rectangle  Width="200" Height="50" Fill="{StaticResource
myBrush}"></Rectangle>
    <Button Margin="10" Width="200" Height="40" Background="{StaticResource
myBrush}" Foreground="White">背景がグラデーションのボタン</Button>
  </StackPanel>
</Window
```



リソースとして定義したブラシ（myBrush）を参照することで、3 つのコントロールでブラシを共有しています。

●その他のリソース

ブラシのほかにも、XML データやスタイルなどもリソースとして利用できます。データ バインディングの回では XML ファイルを読み込んで ListBox に表示しましたが、その際に XML データをリソースとして扱いました。

```
<Window.Resources>
  <XmlDataProvider x:Key="BookData" Source="bookdata.xml"
XPath="Inventory/Books"/>
</Window.Resources>
```

ここでは XML データを提供する XmlDataProvider をリソースとして定義していますが、このように XAML では、さまざまな内容をリソースとして利用できます。

■リソースの有効範囲

ここまでの例では、<Window> 要素の Resources プロパティとしてリソースを記述しました。しかし、各コントロールにも <Window> 要素と同様に Resources プロパティが用意されています。このため、パネルや、Button などのコントロールでもリソースの定義が可能です。

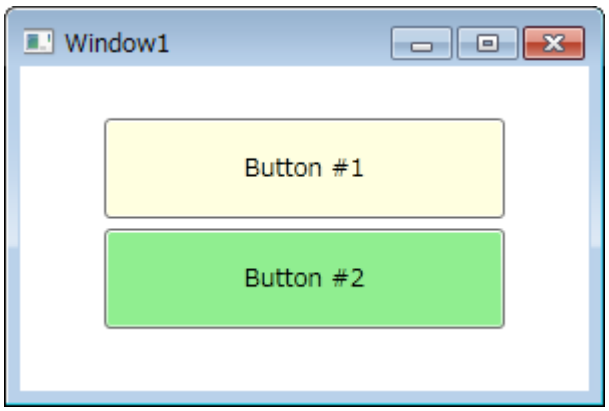
ただしその場合、リソースが参照できるのは、リソースを定義した要素の下位にある要素（コントロール）のみになります。リソースを利用する主な目的は、リソースを共有することにより無駄を省くことですから、通常はルート要素である <Window> 要素にリソースを記述します。

次のリストでは、<Window> 要素で単色のブラシをリソースとして定義し、さらに StackPanel に含まれる <Grid> 要素において、同じキー名で異なる色のブラシを定義しています。

```
<Window x:Class="WpfApplication5.Window1"
```

```
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="Window1" Height="200" Width="300">
  <Window.Resources>
    <SolidColorBrush x:Key="myBrush" Color="LightYellow" />
  </Window.Resources>
  <StackPanel VerticalAlignment="Center">
    <Button Width="200" Height="50" Background="{StaticResource myBrush}">Button
#1</Button>
    <Grid Margin="5">
      <Grid.Resources>
        <SolidColorBrush x:Key="myBrush" Color="LightGreen" />
      </Grid.Resources>
      <Button Width="200" Height="50" Background="{StaticResource
myBrush}">Button #2</Button>
    </Grid>
  </StackPanel>
</Window>
```

実行すると、Grid の中に配置されたボタン（Button #2）は、Grid で定義されたブラシにより背景が描画されるのが分かります。



●アプリケーション全体で適用されるリソース

ウィンドウ単位ではなく、アプリケーション全体でリソースを共有することも可能です。この場合は、アプリケーションの定義を行っている XAML ファイルにリソースを記述します。

Visual Studio 2008 でプロジェクトを新規作成した場合には、自動作成された App.xaml に <Application.Resources> 要素が記述されています。ここに記述したリソースはアプリケーション全体で利用できます。

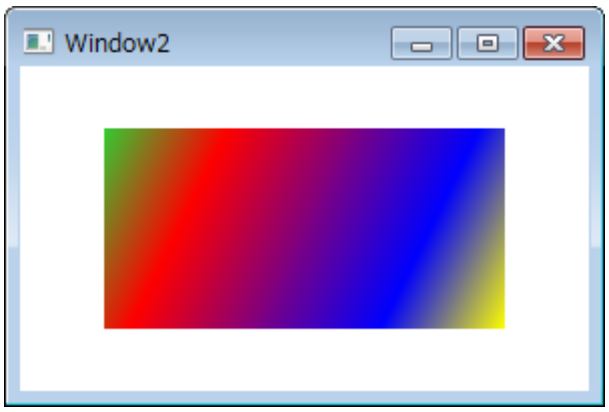
```
<Application x:Class="WpfApplication5.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             StartupUri="Window2.xaml">
```

```
  <Application.Resources>
    <LinearGradientBrush x:Key="myGlobalBrush" StartPoint="0,0" EndPoint="1,1">
      <GradientStop Color="LimeGreen" Offset="0.0" />
      <GradientStop Color="Red" Offset="0.25" />
      <GradientStop Color="Blue" Offset="0.75" />
      <GradientStop Color="Yellow" Offset="1.0" />
    </LinearGradientBrush>
  </Application.Resources>
</Application>
```

このリソースを別の XAML ファイルで参照する場合にも、<Window> 要素で定義したリソースと同様に、リソース名で参照できます。

```
<Window x:Class="WpfApplication5.Window2"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       Title="Window2" Height="200" Width="300">
  <Grid>
    <Rectangle  Width="200" Height="100" Fill="{StaticResource
myGlobalBrush}"></Rectangle>
  </Grid>
</Window>
```

リスト App.xaml で定義したリソースを参照する



■まとめ

今回は、リソースについて学習しました。リソースを記述する場合は、Resources プロパティで 各リソースに x:Key 属性を付加し、リソースを参照する場合は、マークアップ拡張機能を用いて、x:Key 属性で設定した名前により参照します。

次回はスタイルについて学習します。