# WPF コントロール

[WPF][Silverlight][.Net][Universal Windows Platform]

# コンテンツモデル

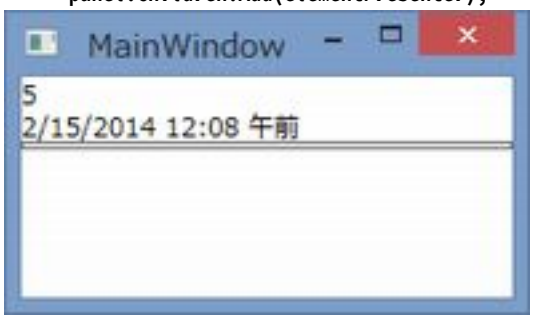
・WPF では、ほとんどのコントロールを 3 つのカテゴリに分類

| カテゴリ | 内容 | 例 |
|---|---|---|
| コンテンツコントロール | 実際の作業を行う他の要素で構成 | ListBox や Button など |
| レイアウトコントロール | 他のコントロールを配置 | StackPanel |
| レンダコントロール | 画面にピクセルを表示する | Rectangle、Ellipse |

## ContentPresenter

・ContentPresenter は Content プロパティに渡されたものを何でも受け取り、それを表示するために対応する視覚ツリーを作成する。
・例えば、ContentPresenter を使用して、数値、日時、ボタンを表示出来る

```
StackPanel panel = new StackPanel();

ContentPresenter intPresenter = new ContentPresenter();
intPresenter.Content = 5;
panel.Children.Add(intPresenter);

ContentPresenter datePresenter = new ContentPresenter();
datePresenter.Content = DateTime.Now;
panel.Children.Add(datePresenter);

ContentPresenter elementPresenter = new ContentPresenter();
elementPresenter.Content = new Button();
panel.Children.Add(elementPresenter);
```



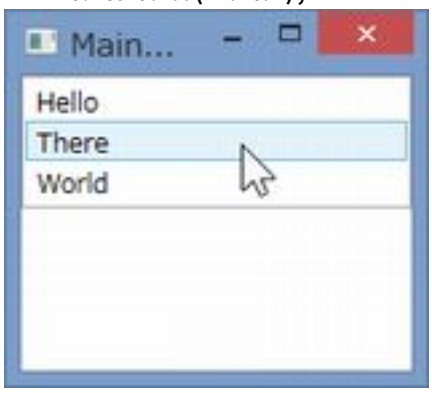

### ■ ContentPresenter がコンテンツを変換して表示するロジック

1.Content が UIElement 型の場合は、表示ツリーに追加
2.ContentTemplate が設定されている場合、それを使用して UIElement インスタンスを生成し表示ツリーに追加
3.ContentTemplateSelector が設定されている場合、それを使用してテンプレートを検索し、そのテンプレートを使用して UIElement インスタンスを生成し表示ツリーに追加
4.Content のデータ型にデータテンプレートが関連づけられている場合、それを使用して UIElement インスタンスを作成
5.Content のデータ型に UIElement 型に変換可能な TypeConverter インスタンスが関連づけられている場合、Content を変換し表示ツリーに追加
6.Content のデータ型に文字列変換可能な TypeConverter インスタンスが関連づけられている場合、Content を TextBlock で包含し表示ツリーに追加
7. 最後に Content の ToString を呼び出し、TextBlock で包含し表示ツリーに追加

### Items

・項目のリストに対しても、ContentPresenter を使用できる
・Items プロパティを使用する

```
ListBox l = new ListBox();
l.Items.Add("Hello");
l.Items.Add("There");
l.Items.Add("World");
```




### Children **および** Child

・UIElement の子のみをサポート

## テンプレート

・WPF コントロールは、プロパティを使用することによる視覚表示の大幅な制御を可能にする
・任意のコントロールの外観を完全にカスタマイズすることも可能。
・カスタマイズの範疇全体を宣言型にし一貫したプログラミングモデルを提供する、テンプレートシステム
・コンテンツコントロール、レイアウトコントロール、レンダコントロールはすべて

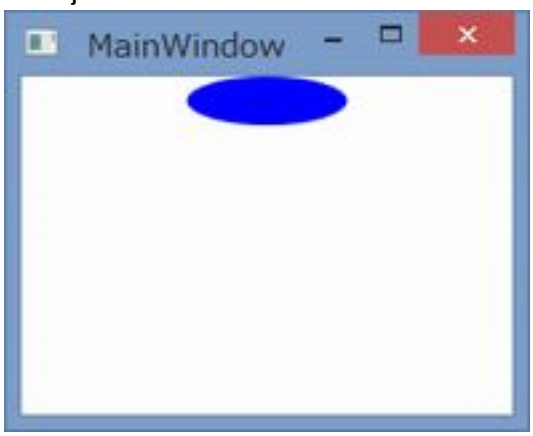
テンプレートをサポート

・コンテンツコントロールは Control から派生し、Template という共通プロパティを継承
・コンテンツに影響を与えずに、外観を変更したい場合新しいテンプレートを定義出来る

```
<Button>
    <Button.Template>
        <ControlTemplate TargetType="{x:Type Button}">
            <Rectangle Fill="Red" Width="75" Height="23"/>
        </ControlTemplate>
    </Button.Template>
    MyButton
</Button>
```


### クリックでテンプレートを変更してみる

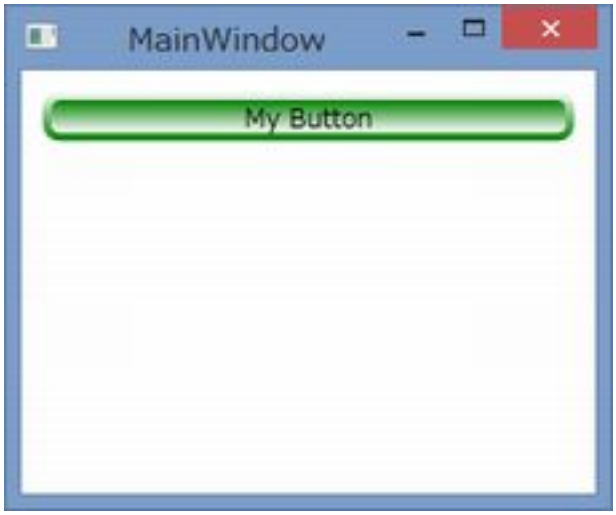
```
private void Button_Click(object sender, RoutedEventArgs e)
{
    var template = new ControlTemplate(typeof(Button));

    // 表示ツリーを定義
    // テンプレートは複数のコントロールに適用できるが、
    // コントロールは表示ツリーで１回しか使用できない
    // ボタンを楕円形で表示させるようにする
    template.VisualTree = new FrameworkElementFactory(typeof(Ellipse));

    // 楕円形が作成されるときに設定されるプロパティリストを作成
    template.VisualTree.SetValue(Ellipse.FillProperty, Brushes.Blue);
    template.VisualTree.SetValue(Ellipse.WidthProperty, 75.0);
    template.VisualTree.SetValue(Ellipse.HeightProperty, 23.0);

    ((Button)sender).Template = template;
}
```




## ボタンらしい外観にしてみる

・コンテンツプレゼンタを含めることでボタンらしい外観のボタンを作成できる
・コンテンツプレゼンタは規定ではテンプレートが適用されるコントロールの Content プロパティの値を表示する

```
<StackPanel Name="MainPanel" Initialized="MainPanel_Initialized" >
    <StackPanel.Resources>
        <ControlTemplate x:Key="ButtonTemplate" TargetType="{x:Type Button}">
            <Border CornerRadius="6" BorderThickness="4">
                <Border.BorderBrush>
                    <LinearGradientBrush EndPoint="0,1">
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Offset="0" Color="White"/>
                            <GradientStop Offset="1" Color="Green"/>
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Border.BorderBrush>
                <Border.Background>
                    <LinearGradientBrush EndPoint="0,1">
                        <LinearGradientBrush.GradientStops>
                            <GradientStop Offset="0" Color="Green"/>
                            <GradientStop Offset="1" Color="White"/>
                        </LinearGradientBrush.GradientStops>
                    </LinearGradientBrush>
                </Border.Background>
                <ContentPresenter
                    HorizontalAlignment="Center"
                    VerticalAlignment="Center"/>
            </Border>
        </ControlTemplate>
    </StackPanel.Resources>
    <Button Template="{StaticResource ButtonTemplate}" Padding="10" Margin="10">
        My Button
    </Button>
</StackPanel>
```




## テンプレートバインディング

・例えばボタンの色を変更するためだけにテンプレートを定義する必要があるとしたら、シンプルなプログラミングモデルとは言えない。
・理想的なのはテンプレートにパラメータを追加できるか、テンプレートコントロールのプロパティをお使用して、テンプレートをカスタマイズできること。

以下の例は、Border の BorderThickness プロパティ、BorderBrush プロパティ、および Background プロパティをテンプレートが適用される Button の同じプロパティにバインドします。Button のプロパティを設定するだけで、下図のようなボタンを作成出来ます。

```
<StackPanel Name="MainPanel" Initialized="MainPanel_Initialized" >
    <StackPanel.Resources>
        <ControlTemplate x:Key="ButtonTemplate" TargetType="{x:Type Button}">
            <Border CornerRadius="6"
                    BorderThickness="{TemplateBinding Property=BorderThickness}"
                    BorderBrush="{TemplateBinding Property=BorderBrush}"
                    Background="{TemplateBinding Property=Background}">
                <ContentPresenter
                    HorizontalAlignment="Center"
                    VerticalAlignment="Center"/>
            </Border>
        </ControlTemplate>
    </StackPanel.Resources>
    <Button Template="{StaticResource ButtonTemplate}"
            Padding="10"
            Margin="10"
            BorderThickness="6"
            BorderBrush="Blue">
        My Button
    </Button>
</StackPanel>
```