CHAPTER 5

Styles and Templates

The most common way for programmers new to XAML to set styles on their controls is with inline styles. In other words, the styling information (font size, font family, etc.) is set as attributes on the control itself, as shown in the following code:

```xml
<Button
    FontSize="24"
    Height="60"
    Width="180"
    Content="Click Me!" />
```

While this works well in small demonstration programs, it gets tedious when setting the same styling information on dozens of controls. Worse, if you decide to modify the style (and you will modify the style!), you must replicate the changes on all of the dozens of controls.

The solution to this problem is to create styles as resources. Styles have a target type and a key, and you can then associate a style with a control with one line of XAML, as shown in the following example:

```xml
<Page.Resources>
    <Style
        TargetType="Button"
        x:Key="ButtonStyle">
        <Setter
            Property="FontSize"
            Value="24" />
        <Setter
            Property="Height"
            Value="60" />
        <Setter
            Property="Width"
            Value="180" />
    </Style>
</Page.Resources>

<Grid
    Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
    <StackPanel>
        <Button
```
You can set as many properties and corresponding values as you like in a style. You could define the style inline, but in this case you’ve defined a reusable style in the Page Resources section. Any button on the page can set its font size, height, and width just by setting its style to this style resource, using the assigned key ButtonStyle. In Figure 5-1, you see an example of two buttons doing just that in the stack panel; they each use the same style and so each have the same width, height, and font size.

![Figure 5-1. Defining styles](image)

You can set as many properties and corresponding values as you like in a style.

**Based on Styles**

You can create new styles based on existing styles, overriding one or more properties and adding new properties. For example, you could create an Urgent button based on the ButtonStyle shown above, but with the letters in yellow, as shown in the following code:

```xml
<Page.Resources>
  <Style TargetType="Button" x:Key="ButtonStyle">
    <Setter Property="FontSize" Value="24" />
    <Setter Property="Height" Value="60" />
    <Setter Property="Width" Value="180" />
  </Style>
  <Style TargetType="Button" x:Key="UrgentButtonStyle" BasedOn="{StaticResource ButtonStyle}">
    <Setter Style="{StaticResource ButtonStyle}" Content="Click Me!" />
    <Setter Style="{StaticResource ButtonStyle}" Content="No! Click Me!" />
  </Style>
</Page.Resources>
```