Support
support@visualcomponents.com

Community
community.visualcomponents.net

Adding styles and themes

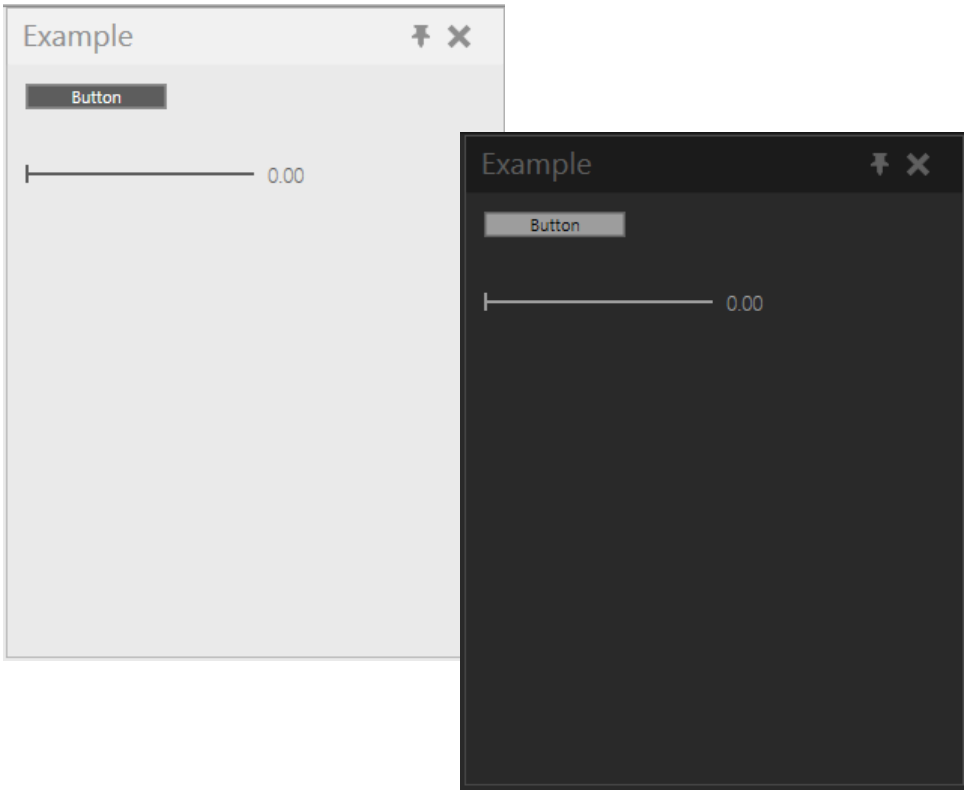
Next Generation | Version: November 20, 2015 | Example: Available upon request

Existing styles and themes in Essentials can be used to personalize the workspace and streamline the design process for extensions.

The setup involves using available resources in Visual Components assemblies to style user controls compatible with every theme in Essentials.

The topics covered in this tutorial include:

- Minimal setup for using resource dictionaries to style controls.
- Creating a custom theme and appearance for Essentials.



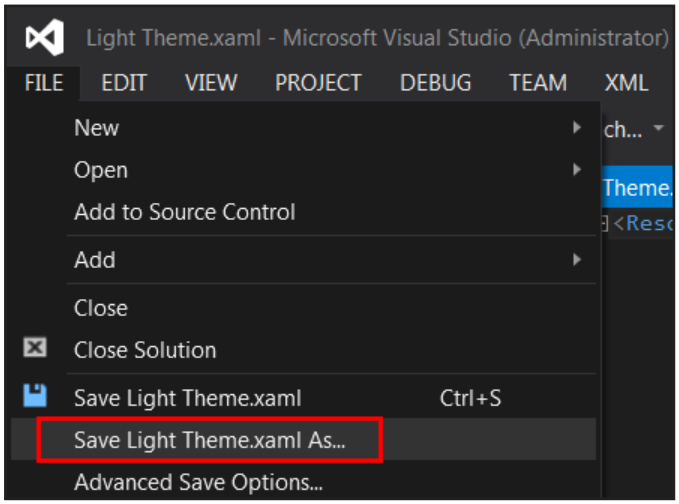
Creating a new theme

A XAML file is used to define a theme in Essentials.

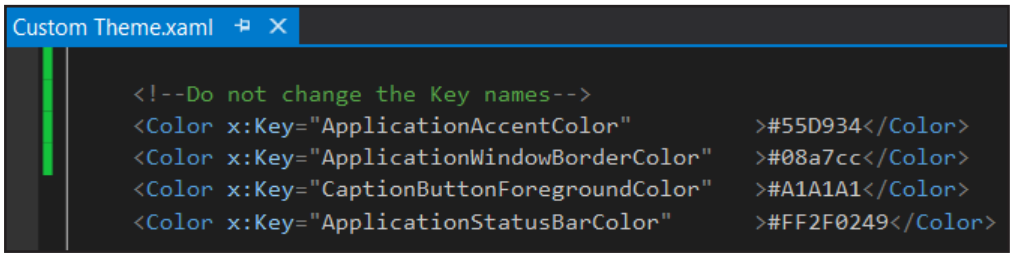
1. Run **Visual Studio** as an administrator in order to write to program files on your device.

2. Open the **Light Theme.xaml** file located in the **Dictionaries** folder of your Essentials program files.

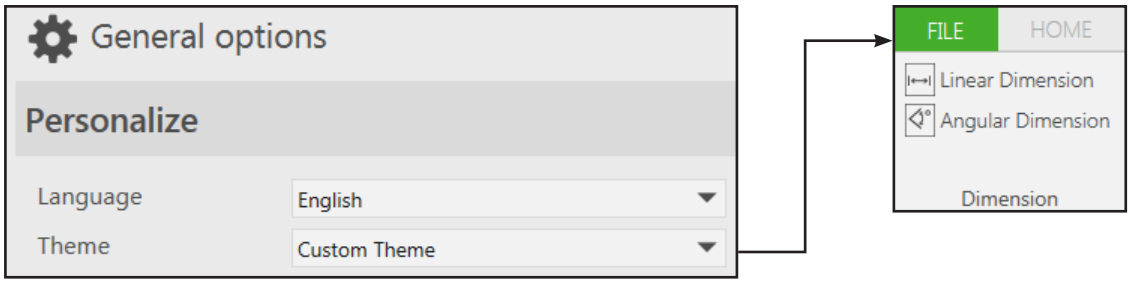
3. Save **Light Theme.xaml** as a new XAML file in the **Dictionaries** folder and use a descriptive filename to represent the name of your theme.



4. Edit the color, brush and font values as needed, and then **save** the file.



5. Run **Essentials** to verify your theme is listed as an option, and then test the design of your theme.



Using a standard set of styles

Resources in the UX.Shared assembly and theme files of Essentials can be used to style an extension.

1. In Visual Studio, create a **Class Library** project.

2. Add references to **Caliburn.Micro**, **System.ComponentModel.Composition**, **System.Xaml** and **UX.Shared**.

NOTE! You can reference the 32-bit version of UX.Shared to verify styles in design-time. If you are using the 64-bit version of UX.Shared, you can verify styles at runtime.

3. Add and compose a **class** that defines a ViewModel for a dockable pane extension of Essentials.

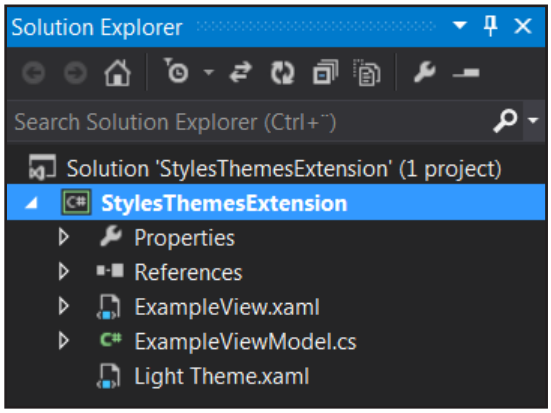
```
namespace StylesThemesExtension
{
    using Caliburn.Micro;
    using System.ComponentModel.Composition;
    using VisualComponents.UX.Shared;

    [Export(typeof(IDockableScreen))]
    public class ExampleViewModel: DockableScreen
    {
        public ExampleViewModel() { this.DisplayName = "Example"; }
    }
}
```

4. Add a **WPF User Control** to represent the View of your ViewModel.

5. In the XAML editor, use a **XML namespace** declaration to map the **UX.Shared** assembly to your View.

```
<UserControl x:Class="StylesThemesExtension.ExampleView"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:shared="clr-namespace:VisualComponents.UX.Shared;assembly=UX.Shared"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
```

6. In your project, add an existing **Essentials theme file** to support brushes used by styles in UX.Shared.



NOTE! A theme file will be used as a design-time resource dictionary and will not be copied in your assembly.

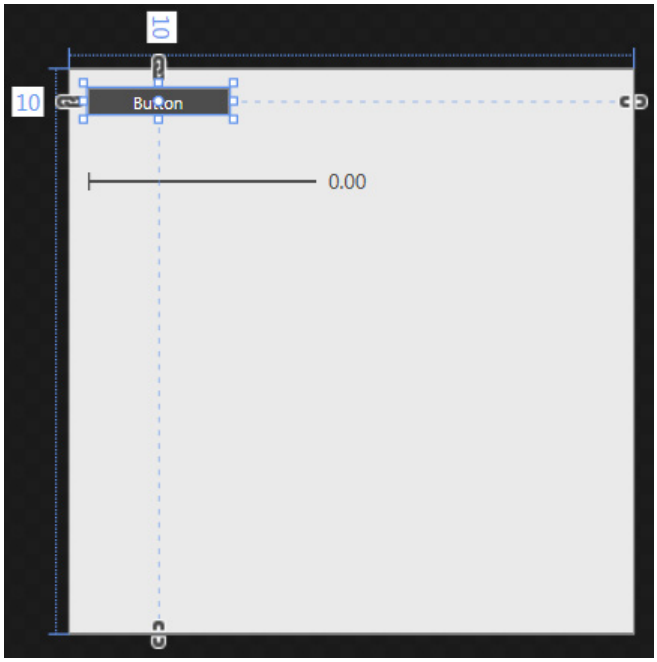
7. In the XAML editor, use a **pack URI scheme** to add and merge resource dictionaries in the root element.

```
<UserControl.Resources>
    <ResourceDictionary>
        <ResourceDictionary.MergedDictionaries>
            <ResourceDictionary Source="pack://application:,,,/UX.Shared;component/Controls/VCThemes/ThemeDictionary.xaml"/>
            <ResourceDictionary Source="pack://application:,,,/UX.Shared;component/Controls/VCThemes/VCGrid.xaml"/>
            <ResourceDictionary Source="pack://application:,,,/UX.Shared;component/Controls/VCThemes/VCButton.xaml"/>
            <ResourceDictionary Source="pack://application:,,,/UX.Shared;component/Controls/VCThemes/VCSliderPlusTextBlock.xaml"/>
            <shared:DesignTimeDictionary Source="Light Theme.xaml"/>
        </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
</UserControl.Resources>
```

IMPORTANT! ThemeDictionary.xaml should be added as a resource in order for the brushes in your extension to update when you change the theme of Essentials.

8. Add and style controls by referring to **keys** in the available resource dictionaries as **dynamic resources**.

```
<Grid Style="{DynamicResource VCSimpleGridStyle}">
    <Button Style="{DynamicResource VCButtonStyle}"
        Content="Button" HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="75"/>
    <Slider Style="{DynamicResource VCSlidePlusTextBlockStyle}"
        HorizontalAlignment="Left" Margin="10,46,0,0" VerticalAlignment="Top" Width="150"/>
</Grid>
```



Verifying styles and themes

You can test your extension during runtime in order to verify the extension is automatically styled and updated based on the active theme in Essentials.

1. Access the properties of your project.

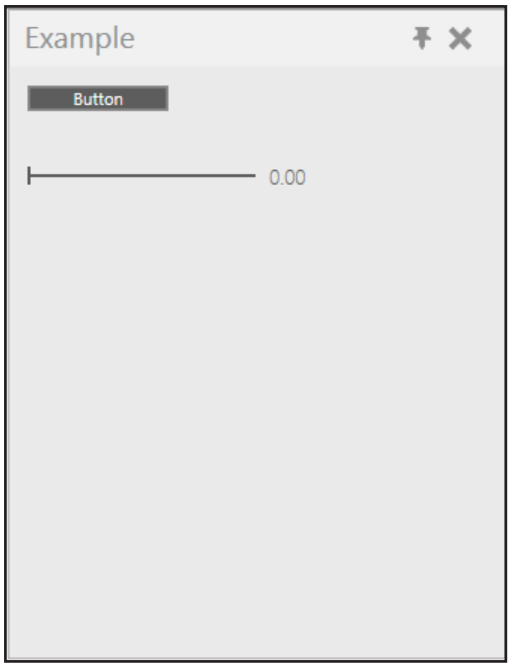
2. In the **Application** tab, set **Assembly name** to have a prefix of **UX.** in order to id your assembly as an Essentials extension.

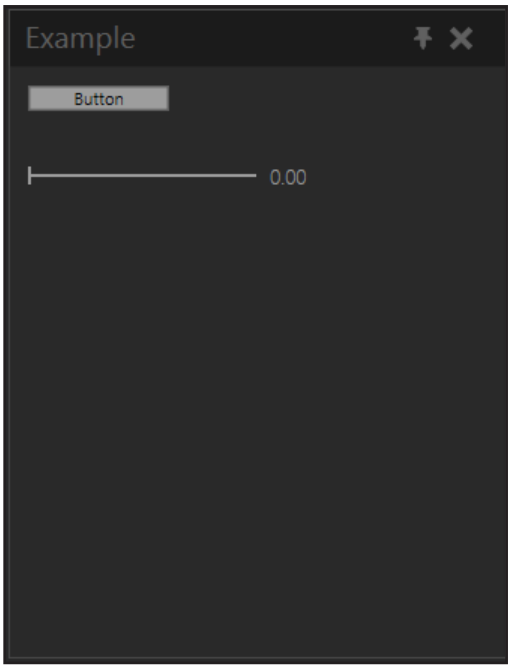
3. In the **Build** tab, set **Platform target** to either **x86** or **x64** depending what version of UX.Shared you referenced in your project, and then set **Output path** to be the path to your Essentials program files.

4. In the **Debug** tab, set **Start external program** to execute your development version of Essentials.

5. Start debugging the application.

6. In **Essentials**, change the **Theme** setting to verify style updates in your extension.





7. Stop debugging the application.

This concludes the tutorial.