telerik
143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

Build Great User Interfaces with the telerik VisualStyleBuilder

by Mike Gunderloy

The telerik RadControls for WinForms suite is more than just a great collection of controls for your .NET Windows Forms user interfaces. It also includes the innovative VisualStyleBuilder utility for customizing those user interfaces. A tool that either developers or designers can use, VisualStyleBuilder lets you perform these tasks completely without code:

- Alter, at design time or run time, any property of any control. Because all telerik controls are composed of primitives, this customization can be applied at a very fine level of detail.
- Inherit property values (such as colors or fonts) from parent elements to child elements, or override them at the child element level.
- Animate changes in response to events, so that a property moves smoothly through a range of values in response to a mouse movement or other event.
- Save themes to be applied consistently across forms or across applications.

You can run the VisualStyleBuilder within Microsoft Visual Studio or as an independent application, allowing you to integrate it flexibly into your workflow. If your development team does their own user interface design, the integrated version is ideal. If you have dedicated designers on your team, the standalone version allows them to work without the complexity of Visual Studio.

Fine-Tuning a Control's Appearance

To see the VisualStyleBuilder in action, start with a RadButton control (telerik's analog of the standard Windows Forms button control) using the default theme, as shown in Figure 1. As with all of the other members of the RadControls suite, the RadButton control is designed to be attractive by default—but it's also easy to customize.

Figure 1. A RadButton control without any customization.

To get started with customization, select Open Theme Builder from the control's Smart Tag menu, as shown in Figure 2. This will launch the VisualStyleBuilder and load the details of the selected control, which you can see in Figure 3.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

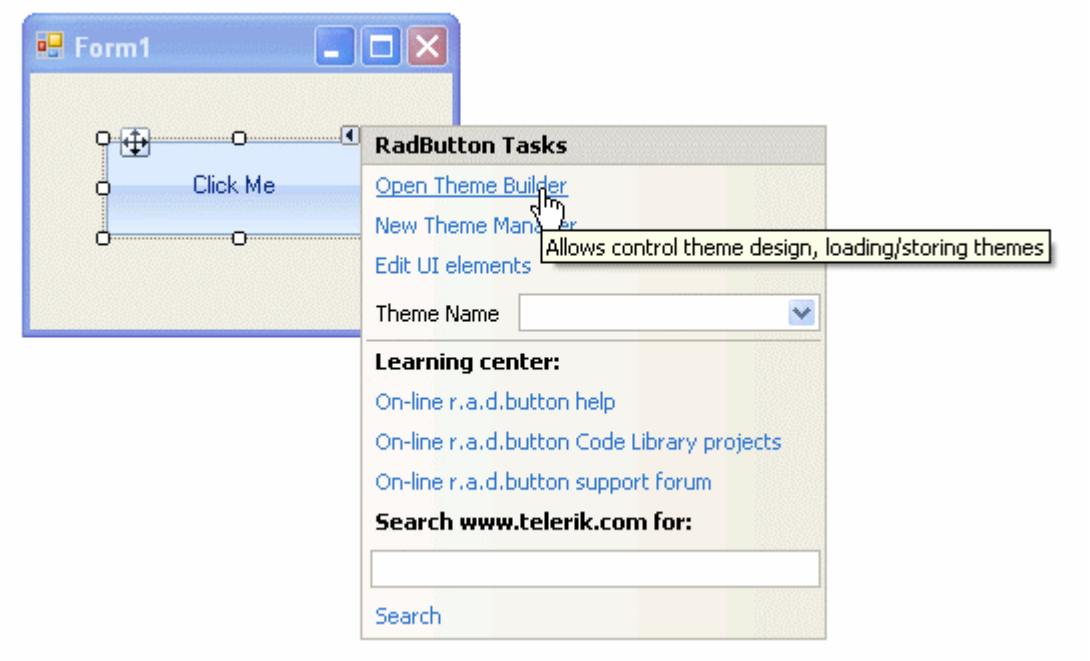

Figure 2. Launching the VisualStyleBuilder.

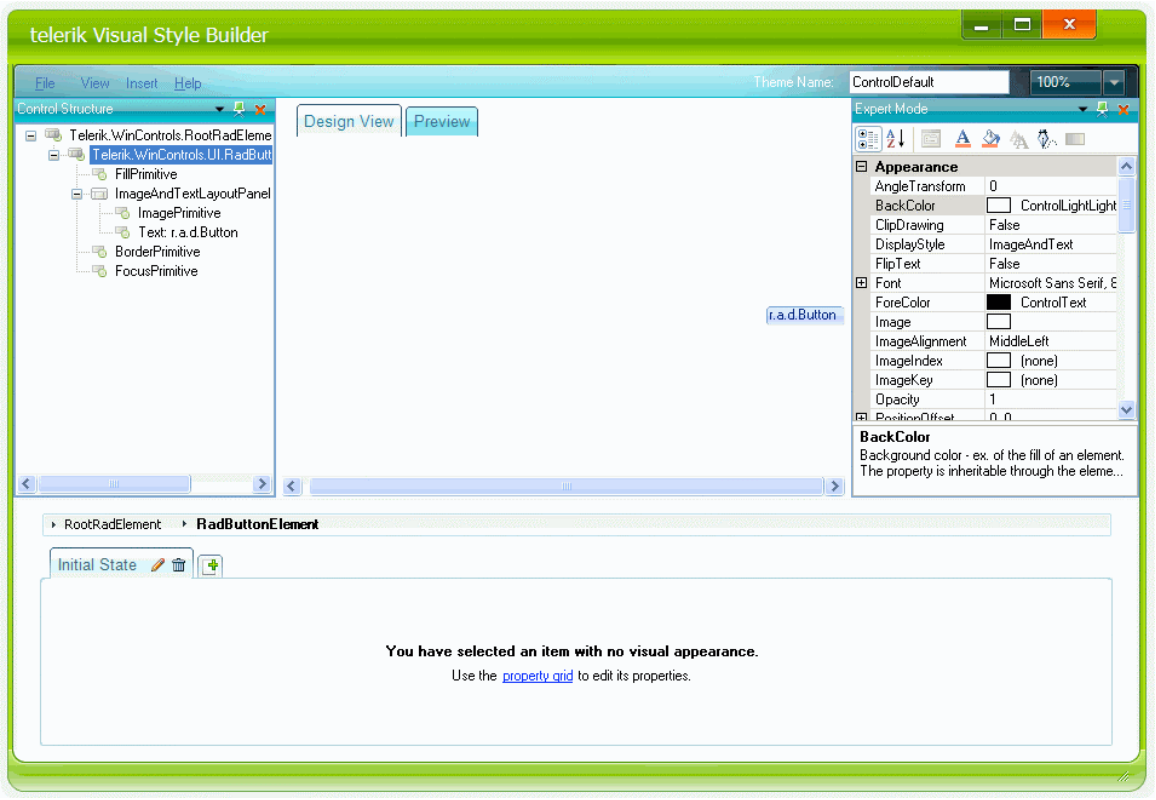

Figure 3. A RadButton loaded into the VisualStyleBuilder.

Let's take a minute to get familiar with this tool (which looks the same whether you launch it from within Visual Studio or directly from the Start menu). telerik's designers have divided VisualStyleBuilder's user interface into five main areas:

- The Theme name textbox at the upper right identifies the theme that you are currently manipulating.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

- The Control Structure tree at the upper left shows the internal structure of the selected telerik RadControl.
- The Control Preview panel in the center shows the effects of applying the current theme to the selected control.
- The State and Property area at the bottom allows you to create states and choose property settings.
- The Expert Mode panel at the upper right allows you to fine-tune property settings.

telerik built the RadControl from a variety of primitives arranged in a logical tree structure. VisualStyleBuilder allows you to navigate this tree and customize the primitives. Any change you make in the tree is immediately reflected in the Control Preview panel.

One thing that makes the VisualStyleBuilder more flexible than the Visual Studio Properties window is the notion of *states*. For example, click on the FillPrimitive node in the Control Element Structure tree of this control and you'll see that there are actually different settings in this theme for various states of the control, as shown in Figure 4.

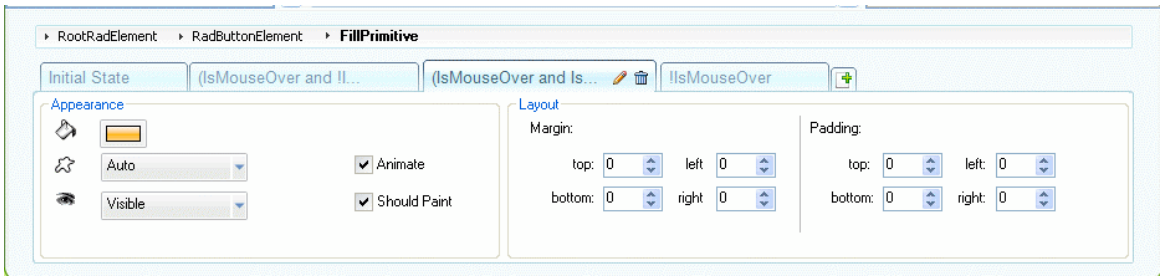

Figure 4. A primitive with multiple states.

In this particular case, we're inspecting the properties of the FillPrimitive (which controls the painting of the background area of the control) that are used when the mouse is over the control and the primary mouse button is clicked, as you can verify by checking the Apply Condition. The red and green dots show property values that the theme sets for this state. Later in this white paper we'll explore the notion of states in more detail.

Editing a Theme

VisualStyleBuilder makes it easy to create your own theme by editing an existing theme. For example, with the state shown in Figure 4 displayed, click on the color patch next to the icon indicating the background fill. This opens the Gradient Editor shown in Figure 5.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

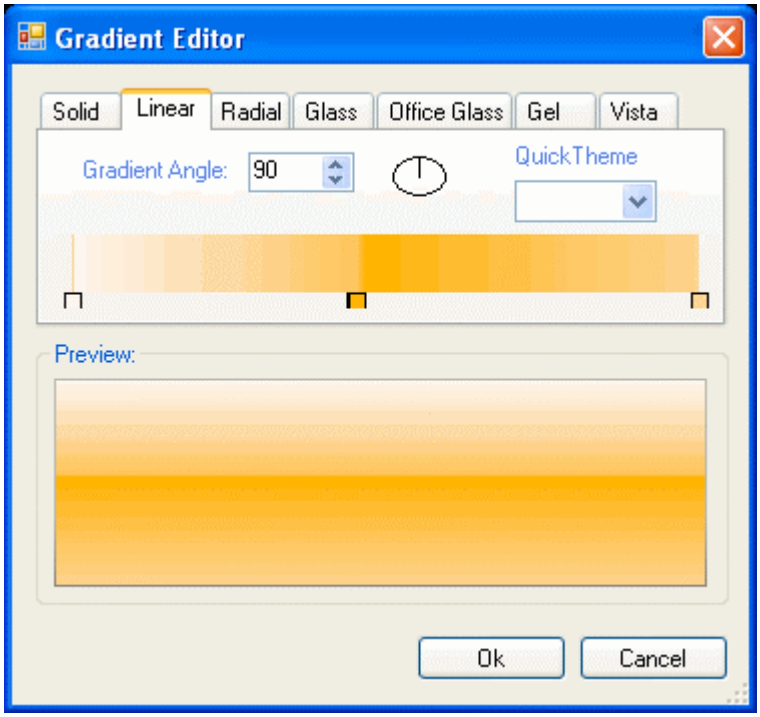

Figure 5. Using the Gradient Editor.

The Gradient Editor provides a visual tool to allow you to set all the parameters associated with a background fill. For example, in the case of a linear gradient, you can slide the small boxes below the color display back and forth to change the relative speed of the change from one color to another, or double-click any box to change the associated area. You can also select any of the seven different gradient styles that the suite supports, from a solid fill to a Windows Vista sparkle.

When you've chosen the colors and other control parameters for your fill, click OK to return to the VisualStyleBuilder. Enter a name for your theme (such as MyControlDefault) and then from the File menu select Save Style Sheet in Theme File. This will allow you to save your custom theme to an XML file that you can later apply to any instance of the RadButton control.

Applying a Theme

The Theme Manager is an invisible component that manages the connection between themes (stored in XML files) and the RadControls on your form. To apply a theme to a control, the form containing the control must also contain a Theme Manager. You can drag and drop a RadThemeManager control from the toolbox to a form, or add one from the Smart Tag menu of any RadControl.

The Smart Tag menu of the Theme Manager itself contains shortcuts for loading themes, as shown in Figure 6.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

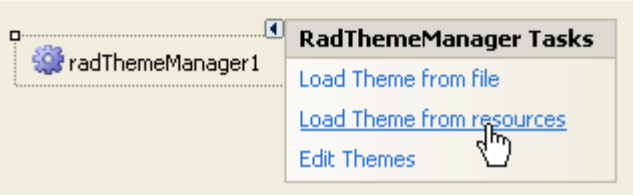

Figure 6. Loading themes via a Theme Manager.

The Theme Manager can load themes from external XML files saved from the VisualStyleBuilder, or from those same XML files saved within your project as embedded resources. In either case, when you load a theme, it becomes available to all controls of the appropriate type through the controls' ThemeName property, as shown in Figure 7.

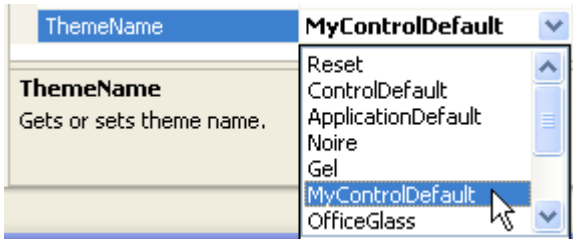

Figure 7. Selecting a theme.

The Theme Manager merges any custom themes that you have loaded with the built-in themes for a particular control to build the list of themes shown in the ThemeName property.

You can also choose to have Themes dynamically loaded and applied at runtime in just a few lines of code. This is useful when you need to build a list of themes from resources that are not available at design time:

```
ThemeSource themeSource1 = new ThemeSource();
themeSource1.StorageType = ThemeStorageType.File;
themeSource1.ThemeLocation = "C:\\MyControlDefault.xml";
radThemeManager1.LoadedThemes.Add(themeSource1);
radButton1.ThemeName = "MyControlDefault";
```

The Telerik Presentation Framework

The simple RadButton control is enough to demonstrate the basics of the VisualStyleBuilder, but there's much more here. The VisualStyleBuilder works hand in hand with the **t**elerik Presentation Framework (TPF). The TPF is the underlying architecture of all controls in the RadControls suite. Any control in the **t**elerik suite is composed of RadElement instances arranged in a logical tree, which you can see in the Control Element Structure pane of the VisualStyleBuilder. For example, Figure 8 shows the structure of a RadMenu control.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

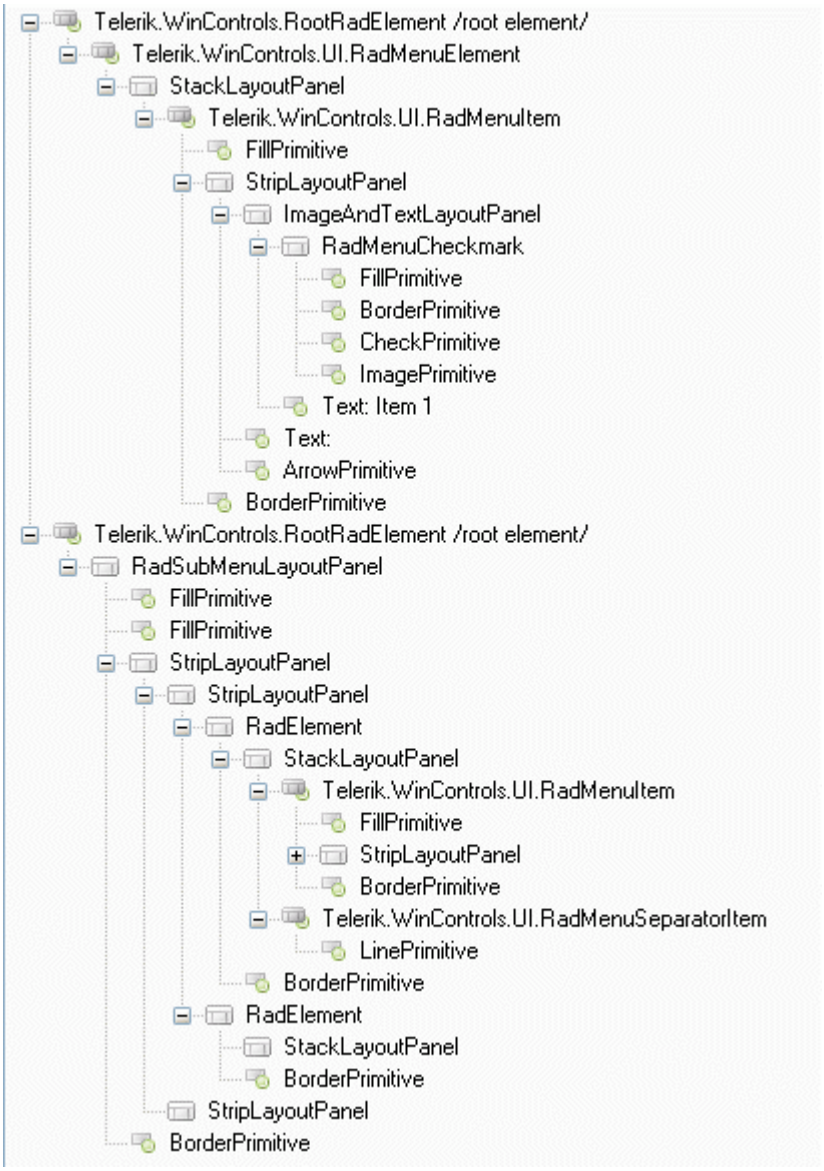

Figure 8. The elements of a RadMenuStrip.

As you can see, the RadMenu control is actually made up of a number of primitives, each of which has its own properties accessible through the VisualStyleBuilder:

- The RadMenuElement supplies properties for the top-level menu as a whole. It uses a StackLayoutPanel to handle the layout of individual main menu items.
- The first RadMenuItem in the tree element controls the overall formatting for main menu items. It includes a FillPrimitive and BorderPrimitive whose properties are inherited by elements lower in the hierarchy.
- The first StripLayoutPanel and ImageAndTextLayoutPanel handle the layout of the individual main menu items and their constituent pieces.
- The RadCheckmark element draws any necessary check marks for checked menu items.
- The FillPrimitive, BorderPrimitive, CheckPrimitive, and ImagePrimitive draw the various pieces of the menu item.
- The two Text elements draw the main text and hint text of the menu items.
- The ArrowPrimitive draws the arrow that indicates a sub-menu, if one is necessary.
- The entire structure then repeats for the sub-menu items, starting with a RadSubMenuLayoutPanel.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

Inheritance in the VisualStyleBuilder

Setting the properties to customize such a structure could quickly become overwhelming, were it not for one of the key features of the TPF: property inheritance. Properties set at a higher level of the control element structure tree are automatically inherited by elements at a lower level of the tree unless you explicitly override their values. For example, if you create a state for the first RadMenuItem element in a RadMenu and set its ForeColor, that value will automatically apply to the children of that element unless you set their ForeColor properties.

This hierarchical property strategy, similar to the way that CSS settings control the display of HTML pages or the way that Microsoft has designed the next-generation controls in the Windows Presentation Foundation, strikes an ideal balance between power and ease of use. If you want to, you can individually set every single property on every single element in a control. But under normal conditions, you need only identify the critical level in the control element structure where an inherited property will have the required effect and set it once to have control-wide consequences.

This approach also makes the RadControls suite an ideal stepping stone on the way from old style Windows Forms controls to next generation user interfaces using Windows Presentation Foundation. By combining hierarchical stylesheet-based customization with a separation of development and design, the RadControls suite will help you understand and apply the new paradigm in a familiar environment.

Managing States

We've mentioned states in passing, but they deserve a more detailed look. The TPF has the capability of attaching specific property settings to any condition of a control that can be detected at runtime, and the VisualStyleBuilder includes a complete interface for defining these states. You can set common states with a single mouse click directly in the VisualStyleBuilder, while a separate Condition Builder lets you handle more complex states including full Boolean logic.

Defining Common States

In many cases, you'll want to define states based on one of a small number of common conditions. The VisualStyleBuilder includes these conditions in a dropdown list, as shown in Figure 9.

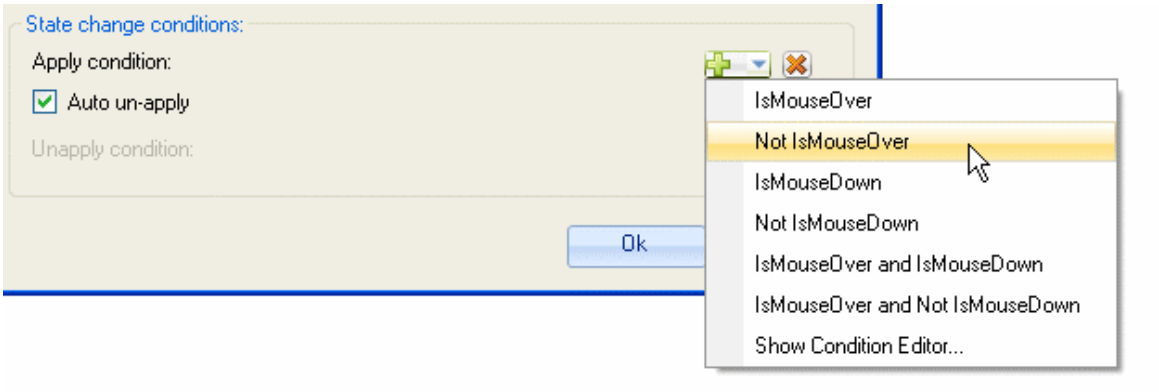

Figure 9. Choosing a common state condition.

You can also choose whether to automatically unapply the state (in which case the unapply condition is the logical opposite of the apply condition), or whether to choose a specific unapply condition.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

Defining Complex States

To define more complex states, you can use the Condition Builder, shown in Figure 10.

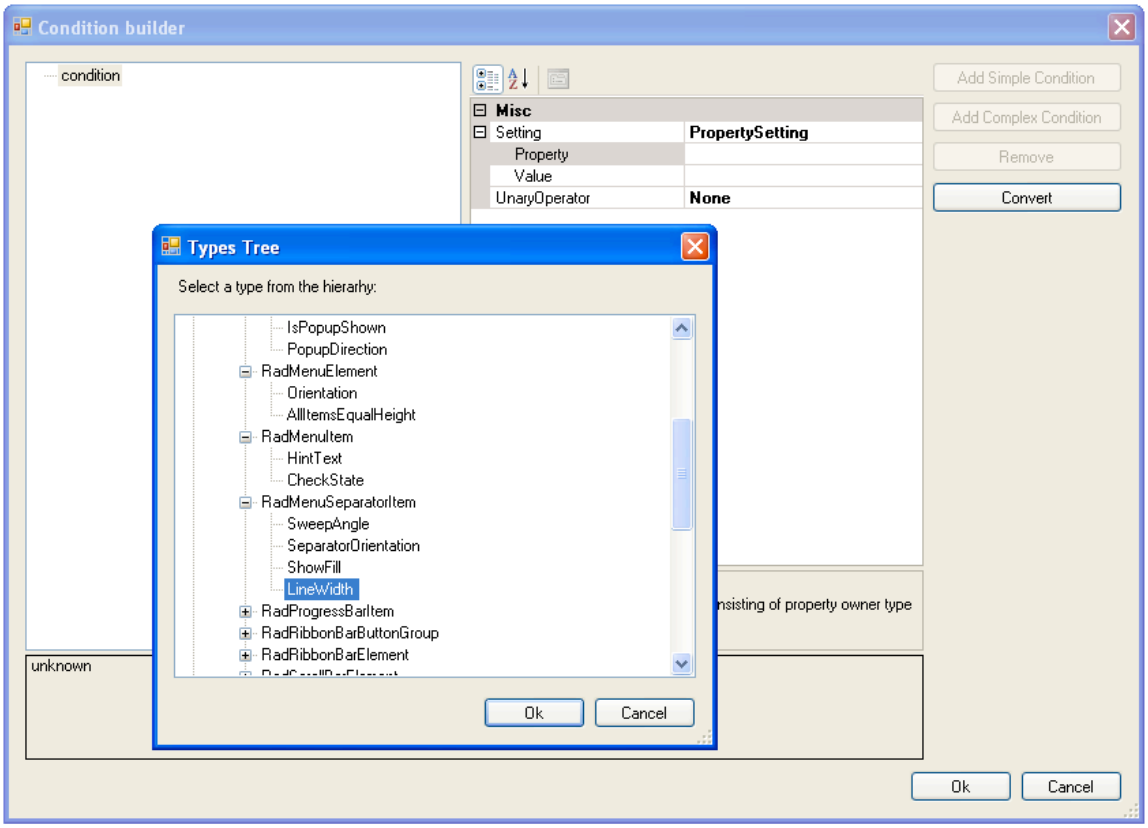

Figure 10. The Condition Builder.

The Condition Builder lets you specify the property and value that define a condition. A type picker lets you select any integer, Boolean, or enumerated property. For example, if you want a state to only apply when the width of the separators on a menu has a particular value, you're free to define it that way. If this isn't enough flexibility for you, the Condition Builder also lets you combine multiple conditions with AND, OR and XOR operators to create compound conditions. And like simple states, complex states can be automatically unapplied or can have specific unapply conditions.

Animating a Property

Another area in which the VisualStyleBuilder goes far beyond previously available control frameworks is in its support for animated properties. For example, suppose you have a button with a normally blue fill that you want to have smoothly fade to a yellow fill when the mouse passes over it, and then return to blue when the mouse leaves the control. With a standard Windows Forms button, this would require writing code, but with the TPF and the VisualStyleBuilder, the ability to animate the change in property values from state to state is a built-in part of the product. Just right-click on the property in the target state (in this case, the IsMouseOver state) and select Convert To Animated.

For more precise control, you can select Animate from any property's shortcut menu to open the Animation Settings dialog box. This dialog box varies according to the type of property being animated; Figure 11 shows the Color Property Animation Settings dialog box.

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

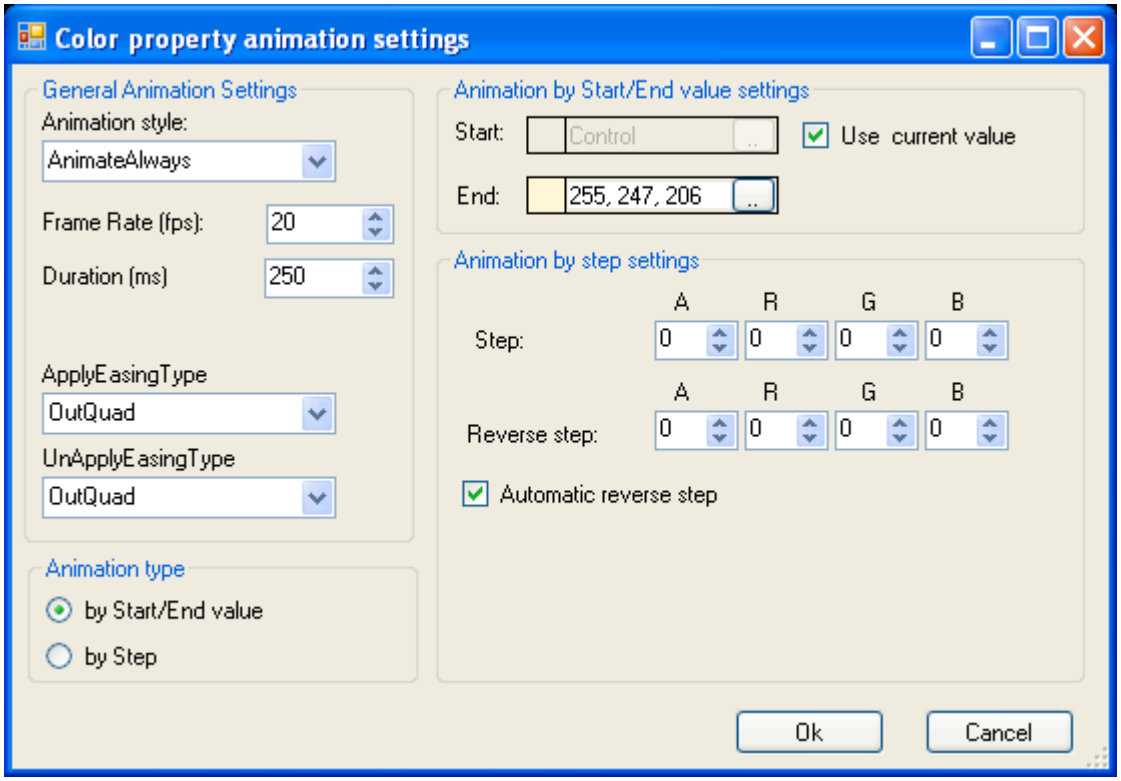

Figure 11. Setting animation properties for a color property.

As you can see, you can control every detail of the animation:

- How long it takes to execute
- What frame rate to use
- The starting and ending property values
- How big a step each frame should represent
- Whether the animation should speed up or slow down at the start or end

You can animate more than color properties using the VisualStyleBuilder. In fact, you can animate any property with a numeric representation. For example, you could make a font size dynamically grow and shrink in response to tab selection, or rotate the text of a menu item when it's checked. The VisualStyleBuilder gives you complete flexibility to use your imagination here.

Building Themes

Finally, the VisualStyleBuilder is an important part of creating and reusing application-wide themes. When you save an individual control's customizations to an XML file, the VisualStyleBuilder includes in the XML file the control type that the customization applies to:

```
<?xml version="1.0" encoding="utf-8" ?>
<XmlTheme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
ThemeName="MyControlDefault">
  <BuilderRegistrations>
    <XmlStyleBuilderRegistration ElementType="Telerik.WinControls.RootRadElement"
RegistrationType="ElementTypeControlType" ElementName=""
BuilderType="Telerik.WinControls.DefaultStyleBuilder"
ControlType="Telerik.WinControls.UI.RadButton" ControlName="">
…
    </XmlStyleBuilderRegistration>
  </BuilderRegistration>
</XmlTheme>
```

143 Brooks Street, Suite 2
Brighton, MA 02135
Phone: 1-866-886-0452, Fax: 1-617-249-2116
info@telerik.com

This particular XML file contains customizations for a RadButton control. But the VisualStyleBuilder can easily merge the customizations for multiple control types into a single XML file, and the ThemeManager will apply only the applicable customizations for a particular control.

Imagine, then, that you're tasked with creating a corporate branding theme for your organization. You might have, for example, a set of colors and fonts chosen by your designers that need to be applied uniformly across all controls and all applications. To do this with the TPF and the VisualStyleBuilder, follow these steps:

1. Open the VisualStyleBuilder and load each control type in turn.
2. Apply the necessary customizations to the control using the VisualStyleBuilder's tools.
3. Save the customized theme to a single XML file.
4. Add a ThemeManager control to each form of every application.
5. Select your corporate theme for every control.

Though creating the initial theme containing the full range of customizations might be time-consuming, this is work that only needs to be done once—and thanks to the standalone version of the VisualStyleBuilder, it doesn't even need to be done by a developer.

Summing Up

The VisualStyleBuilder is just one part of the **t**elerik RadControls for WinForms suite (which, after all, also includes a variety of cutting-edge Windows Forms controls). But it's a key part of the suite, enabling all of the capabilities that you've learned about in this white paper:

- Fine-grained control over all parts of a control's appearance
- Design-time or run-time theme support
- Inherited hierarchical property values
- Complex state-based property management
- Animated property transitions
- Application-wide themes

To see the VisualStyleBuilder in action for yourself, you can review the online demos (http://www.telerik.com/support/videos.aspx) or download a free trial of the suite. Visit us at http://www.telerik.com/products/winforms/overview.aspx to learn more.