Silverlight 3 Controls

For those who have worked with Silverlight 1.0, one of the first observations you most likely made was the lack of common controls such as the Button, TextBox, and ListBox. In fact, Silverlight 1.0 provided only two basic controls: Rectangle and TextBlock. From these, the developers were expected to implement all of the rich controls they needed. As you can imagine, it was quite a bit of work to create all of the form controls using just these two base controls.

Since then, Microsoft’s vision of Silverlight has gone beyond basic animations to spark up your applications and into the realm of feature-rich user interfaces (UIs). To this end, Silverlight 3 includes a strong base of controls that you can use within your Silverlight applications.

In this chapter, you will first look at the Silverlight controls in general by examining control properties and events. You will then take a brief tour of some of the more common form controls included in Silverlight 3. This chapter is meant to provide a high-level introduction to these common Silverlight controls. You will continue to work with the controls throughout the remainder of the book, so you will see more specific usage scenarios.

Setting Control Properties

The most straightforward and simple way to set a property is by using attribute syntax. However, in some cases, you will use element syntax.

Attribute Syntax

Most properties that can be represented as a simple string can be set using attribute syntax. Setting an attribute in XAML is just like setting an attribute in XML. An XML element contains a node and attributes. Silverlight controls are defined in the same way, where the control name is the node, and the properties are defined as attributes.

As an example, you can easily use attribute syntax to set the Width, Height, and Content properties of a Button control, as follows:

<Button Width="100" Height="30" Content="Click Me!"></Button>
Element Syntax

Element syntax is most commonly used when a property cannot be set using attribute syntax because the property value cannot be represented as a simple string. Again, this is very similar to using elements in XML. The following is an example of setting the background color of a button:

```xml
<Button Width="100" Height="30" Content="Click Me!">
    <Button.Background>
        <SolidColorBrush Color="Blue"/>
    </Button.Background>
    <Button.Foreground>
        <SolidColorBrush Color="Red"/>
    </Button.Foreground>
</Button>
```

Type-Converter-Enabled Attributes

Sometimes when defining a property via an attribute, the value cannot be represented as a simple string—rather, it is converted to a more complex type. A common usage of a type-converter-enabled attribute is Margin. The Margin property can be set as a simple string, such as in the following:

```xml
<Button Width="100" Content="Click Me!" Margin="15"></Button>
```

When you set the Margin property in this fashion, the left, right, top, and bottom margins are all set to 15 pixels. What if you want to set the top margin to 15 pixels, but you want the other three margins to be 0? In order to do that, you would set the Margin property as follows:

```xml
<Button Width="100" Content="Click Me!" Margin="0,15,0,0"></Button>
```

In this case, Silverlight takes the string "0,15,0,0" and converts it into a more complex type. The string is converted to four values: left margin = 0, top margin = 15, right margin = 0, and bottom margin = 0.

This type-conversion concept is not new to Silverlight. For those of you familiar with Cascading Style Sheets (CSS), the same sort of structure exists. As an example, when you are defining a border style, within the simple string value for a border, you are actually setting the thickness, color, and line style. The following border assignment in CSS will set the border thickness to 1 pixel, the line style to be solid, and the color to #333333 (dark gray):

```css
border: 1px solid #333333;
```

Attached Properties

In Chapter 3, you learned how to set a control’s position within a Canvas panel by using attached properties. An attached property is a property that is attached to parent control. In the Chapter 3’s example, you specified the Button control’s position within the Canvas object by setting two attached properties: Canvas.Top and Canvas.Left. These two properties reference the Button control’s parent, which is the Canvas.

```xml
<Canvas>
    <Button Width="100" Content="Click Me!"
        Canvas.Top="10" Canvas.Left="13" />
</Canvas>
```