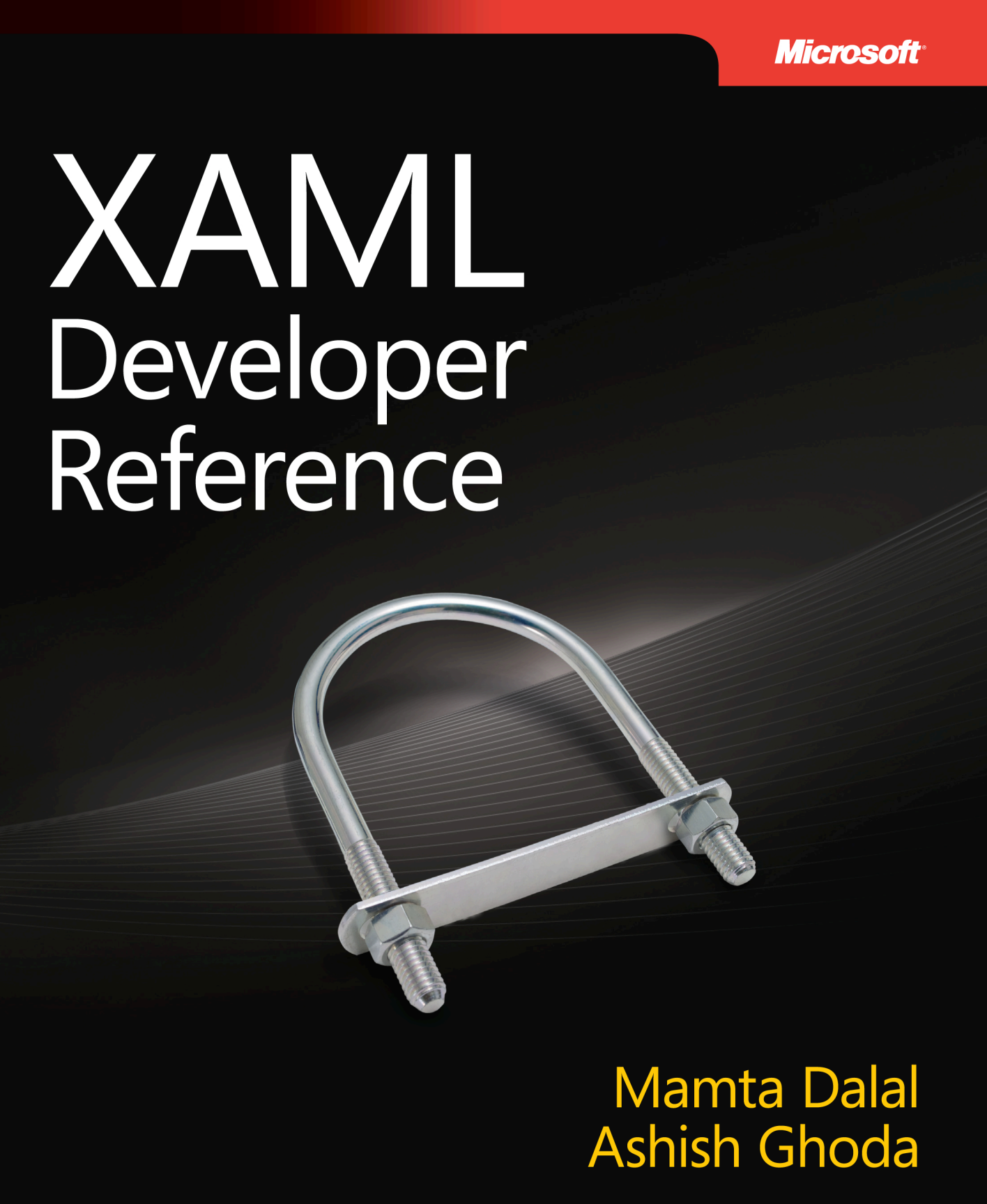
Microsoft

XAML Developer Reference



Mamta Dalal
Ashish Ghoda

XAML Developer Reference

Your expert guide to designing and building dynamic user interfaces

Sharpen your application design and development skills using XAML—the declarative markup language used in Microsoft® Silverlight®, Windows® Presentation Foundation (WPF), and the Windows 8 Runtime APIs. Led by two XAML experts, you'll learn practical ways to build rich, interactive user interfaces with data integration capabilities and support for multimedia, graphics, and animation. This hands-on guide is ideal for Microsoft .NET developers and web designers alike.

Discover how to:

- Control UI behavior and implement business logic with code-behind solutions
- Manage UI element positioning with the XAML layout system
- Use templates to customize UI elements—without affecting their functionality
- Apply different types of property and event systems in WPF and Silverlight
- Bind various kinds of data to your UI, and display them in the format you want
- Implement 2D and 3D vector graphics and animations
- Reuse control styles and properties to maintain consistency throughout your application

Get code and project samples on the web
Ready to download at
http://go.microsoft.com/FWLink/?Linkid=233593

*For **system requirements**, see the Introduction.*


About the Authors

Mamta Dalal has more than 10 years of experience developing Windows and web applications. She is an active contributor to the .NET community and has written several articles on C#, .NET, Silverlight, and WPF.

Ashish Ghoda, founder and president of a technology consulting firm, is an accomplished author with more than 15 years of experience in enterprise architecture, application development, and technical and financial management.

RESOURCE ROADMAP

Developer Step by Step
- Hands-on tutorial covering fundamental techniques and features
- Practice exercises
- Prepares and informs new-to-topic programmers

Developer Reference
- Expert coverage of core topics
- Extensive, pragmatic coding examples
- Builds professional-level proficiency with a Microsoft technology

Focused Topics
- Deep coverage of advanced techniques and capabilities
- Extensive, adaptable coding examples
- Promotes full mastery of a Microsoft technology

See inside cover

ISBN: 978-0-7356-5896-7

microsoft.com/mspress

U.S.A. $39.99
Canada $41.99
[*Recommended*]

Programming/Windows/
Microsoft Visual Studio

Windows®

Microsoft®

```
        (DependencyObject d, RoutedEventHandler hander)
    {
        UIElement element = d as UIElement;
        if (element != null)
        {
            element.RemoveHandler(MyWindow.MyCustomAttachedEventEvent, handler);
        }
    }
}
```

The *EventSetter* and *EventTrigger* Classes

As discussed earlier, XAML enables you to set up a common set of styles targeted to specific types of controls. But just as you can use styles and templates to set common set of properties to provide a consistent, unified look, XAML also lets you set up a common event handler that corresponds to a specific event. This event handler can perform unified actions (including the use of the *Storyboard* element to provide some types of animation) using the *EventSetter* and *EventTrigger* classes and their corresponding XAML elements.

The *EventSetter* Class

Supported By	
WPF	Yes
Silverlight	No

In XAML, the *EventSetter* represents a specific event handler that should be invoked in response to a corresponding routed event. Within the scope of the *Style*, you can set *EventSetter* as an object element in the XAML file and define the event hander for the targeted control (such as the *Click* event for a *Button* control).

Earlier in this chapter, in the section titled "The *RoutedEventArgs* Class," you saw how to set up a common *Click* event called *CommonButtonClickEvent* for the *ButtonBase* class at the *StackPanel* level by attaching the *ButtonBase.Click* in XAML to the *StackPanel* control. You can achieve the same functionality by adding a style setter at the *StackPanel* level.

Reopen the WpfApplication1 project, open the MainWindow.xaml page, and locate the *StackPanel* element in the XAML code. The *StackPanel* contains two buttons: Submit and Cancel. Remove the *ButtonBase.Click="CommonButtonClickEvent"* from the *StackPanel* element and add the highlighted portion (in bold text) from the following code. These are *StackPanel* resources that create the *EventSetter* as a style targeted toward the *Click* event of *Button* controls (in this case targeting the Submit and Cancel buttons) and process the *CommonButtonClickEvent* event handler method when a *Click* event is raised:

```
<StackPanel
    Orientation="Horizontal"
    HorizontalAlignment="Right"
    Grid.Column="1"
    Grid.Row="3">
    <StackPanel.Resources>
```

```
        <Style TargetType="{x:Type Button}">
            <EventSetter
                Event="Click"
                Handler="CommonButtonClickEvent"/>
        </Style>
    </StackPanel.Resources>
    <Button
        x:Name="SubmitButton"
        Content="Submit"
        Margin="5"
        Width="100"
        Click="SubmitButton_Click"/>
    <Button
        x:Name="CancelButton"
        HorizontalAlignment="Left"
        Content="Cancel"
        Margin="5"
        Width="100"/>
</StackPanel>
```

If you now compile and run the project, you should get similar results as you did before when clicking the Submit and Cancel buttons. (See the results discussed on page 74, in the section "The *RoutedEventArgs* Class.")

Caution The *EventSetter* is applicable to only routed events and is available only for the WPF framework. Silverlight does not support *EventSetter*.

The *EventTrigger* Class

Supported By	
WPF	Yes
Silverlight	Yes

The *EventTrigger* class lets you set actions and apply animations in response to specific routed events. Event triggers make use of the *Storyboard* element to apply an animation. You can declare an *EventTrigger* within *Style* or page-level elements as part of the *Triggers* collection, or in a *ControlTemplate*.

The following XAML code snippet defines an *EventTrigger* for the *Rectangle.Loaded* event as part of the *Rectangle.Triggers* collection. The event enables you to change the color of the *Rectangle* when it gets loaded. The *EventTrigger* property has an attribute called *RoutedEvent* that indicates which event will trigger the action:

```
<Rectangle.Triggers>
    <EventTrigger
        RoutedEvent="Rectangle.Loaded">
        <BeginStoryboard>
            <Storyboard>
                <ColorAnimation
                    Storyboard.TargetName="brush"
                    Storyboard.TargetProperty="Color"
```

```
                    To="Magenta"
                    Duration="0:0:6"/>
            </Storyboard>
        </BeginStoryboard>
    </EventTrigger>
</Rectangle.Triggers>
```

More Info See Chapter 5 for more information on resources and styling. See Chapter 9, "Media, Graphics, and Animation," for more information on animation.

The next chapter, "Markup Extensions and Other Features," covers markup extensions that can extend the capabilities of XAML to resolve property values at runtime. It also covers XAML services and security measures in XAML.

Summary

This chapter introduced the new dependency property and routed event systems for XAML and the WPF and Silverlight frameworks.

A dependency property is backed by a regular CLR .NET property and uses the dependency property framework to determine the property value based on various possible sources, including data binding, animation, template resources specified in the XAML, styles, or local values. Dependency properties also provide a value change notification service.

Attached properties are dependency properties that provide dynamic extension of classes without inheritance, and relate child objects to parent objects in a predefined specific context.

Routed events are CLR-type events backed by the *RoutedEvent* class and processed by the WPF event system. You can define a routed event within a XAML file using attribute syntax or implement a routed event in code-behind. Routed events follow one of three possible routing paths: bubble up, tunneling, or direct routing. A routed event can have one object element as the event sender and one or more event receivers (which may include the object that raises the event) that can execute an event handler implemented in code-behind. Like attached dependency properties, attached routed events are a special type of routed event that is specific to XAML and not wrapped by .NET CLR add and remove handler accessors. The attached event is neither owned by event sender nor by event receiver. You can attach an attached routed event arbitrarily to any object element.

PART 2

Enhancing User Experience