ComponentOne

Book for WPF

ComponentOne, a division of GrapeCity

201 South Highland Avenue, Third Floor

Pittsburgh, PA 15206 • USA

Internet: info@ComponentOne.com

Web site: http://www.componentone.com

Sales

E-mail: sales@componentone.com

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

This manual was produced using ComponentOne Doc-To-Help™.

Table of Contents

ComponentOne Book for WPF Overview

Present information using a familiar book metaphor with **C1Book**, a page-turning book control for innovative WPF navigation. With the **C1Book** control you can present **UIElement** objects as if they were pages in a regular paper book. You can see two elements at a time, add shadows, turn pages with the mouse, and more with **ComponentOne Book™ for WPF**.

For a list of the latest features added to **ComponentOne Studio for WPF**, visit What's New in Studio for WPF.

Getting Started

Get started with the following topics:

Help with ComponentOne Studio for WPF

Getting Started

For information on installing **ComponentOne Studio for WPF**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with Studio for WPF.

What's New

For a list of the latest features added to **ComponentOne Studio for WPF**, visit What's New in Studio for WPF.

Key Features

ComponentOne Book for WPF allows you to create customized, rich applications. Make the most of **Book for WPF** by taking advantage of the following key features:

- **Familiar Book Metaphor**

  C1Book enables you to present information innovatively using a familiar mental model – that of a book. But **Book for WPF** is not a typical static book, it's dynamic and interactive, and it takes the familiar metaphor further using WPF.

- **Real Book-like Visuals**

  C1Book enables you to customize the look and feel of the book pages; for example, show page folds and display inner and outer shadows. It not only looks like a book, but can be interacted with like a book.

- **Flexible Data Binding**

  C1Book is an **ItemsControl**, so you can bind it to any data source. Each item in the data source can be a **UIElement** or a generic object that gets converted into a **UIElement** using templates.

- **Custom Styles for Book Pages and Cover**

  C1Book supports different templates for odd and even pages, and it is possible to define custom pages like the cover.

Book for WPF Quick Start

The following quick start guide is intended to get you up and running with **Book for WPF**. In this quick start you'll start in Visual Studio and create a new project, add a **Book for WPF** control to your application, and customize the appearance and behavior of the control.

You'll create a WPF application using a C1Book control that contains a variety of content, you'll add a C1Book control to the application, customize and add content to the Book, and observe some of the run-time interaction possible with **Book for WPF**.

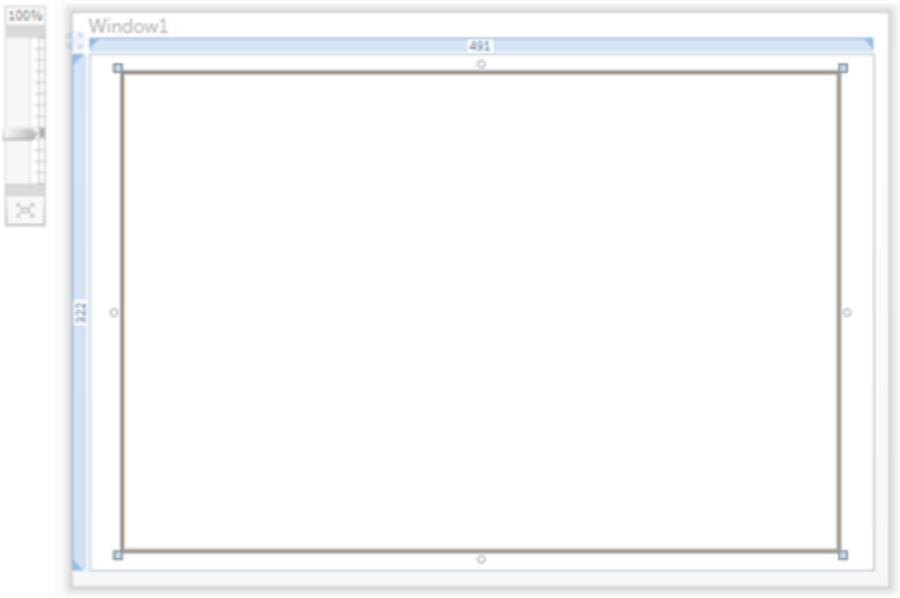
Step 1 of 3: Creating the Book Application

In this step you'll create a WPF application using **Book for WPF**. When you add a C1Book control to your application, you'll have a complete, functional book-like interface that you can add images, controls, and other elements to. To set up your project and add a C1Book control to your application, complete the following steps:

1. Create a new WPF project in Visual Studio. For more information about creating a WPF project, see Creating a .NET Project in Visual Studio.
2. In the Solution Explorer, right-click the **References** item and choose **Add Reference**. Select the **C1.WPF**, **C1.WPF.Extended**, and **WPFToolkit** assemblies and click **OK** to add references to your project.
3. Navigate to the Visual Studio Toolbox, and double-click the **C1Book** icon to add the control to the window.
4. Resize the window and reposition the **C1Book** control in the window.
5. Click once on the **C1Book** control in Design view, navigate to the Properties window, and set the following properties:
   - Set **Width** to "450" and **Height** to "300".
   - Set **HorizontalAlignment** and **VerticalAlignment** to **Center** to center the control in the panel.
   - Check the **IsFirstPageOnTheRight** check box to set the first page to appear on the right side.
   - Set the **TurnInterval** property to 600 to increase the time taken for the page turn animation.

   The XAML will appear similar to the following:
   ```
   <c1:C1Book Name="C1Book1" Width="450" Height="300"
   VerticalAlignment="Center" HorizontalAlignment="Center"
   IsFirstPageOnTheRight="True" TurnInterval="600" />
   ```
6. Update the **C1Book** control's markup in XAML view, so that it includes a closing tag as in the following:
   ```
   <c1:C1Book Name="C1Book1" Width="450" Height="300"
   VerticalAlignment="Center" HorizontalAlignment="Center"
   IsFirstPageOnTheRight="True" TurnInterval="600"></c1:C1Book>
   ```

   The page's Design view should now look similar to the following image (with the C1Book control selected on the form):



You've successfully set up your application's user interface, but the C1Book control currently contains no content. In the next step you'll add content to the C1Book control, and then you'll add code to your application to add functionality to the control.

Step 2 of 3: Adding Content to the Book Control

In this step you'll add content to the C1Book control in design-time, XAML markup, and code. You'll add standard Microsoft controls and content to create a virtual book with several pages that can be turned. To customize your project and add content to the C1Book control in your application, complete the following steps:

1. Click once on the C1Book control to select it.
2. Navigate to the Toolbox and double-click the **TextBlock** control to add it to the project.
3. In XAML view move the **TextBlock**'s tag so it is within the C1Book control's tags.
4. Select the **TextBlock** in Design view, navigate to the Properties window, and set the following properties:
   - **Text** to "Hello World!"
   - **HorizontalAlignment** to **Center**
   - **VerticalAlignment** to **Center**
5. Switch to XAML view and add two button controls in the markup just after the **TextBlock**. The markup will appear like the following:

```
<c1:C1Book Name="C1Book1" Width="450" Height="300"
VerticalAlignment="Center" HorizontalAlignment="Center"
IsFirstPageOnTheRight="True" TurnInterval="600">
    <TextBox Height="23" HorizontalAlignment="Center"
Margin="10,0,0,102" Name="TextBox1" VerticalAlignment="Center"
Width="120">Hello World!</TextBox>
    <Button x:Name="Button1" Content="Last" Height="100" Width="100"
Click="Button1_Click"/>
    <Button x:Name="Button2" Content="Next" Width="150" Height="150"
Click="Button2_Click"/>
</c1:C1Book>
```

This will give the buttons names so they are accessible in code, size the controls, and add event handlers that you will add code for in the next steps.

6. In Design view, double-click the window to switch to Code view.
7. In Code view, add the following import statements to the top of the page:

   - Visual Basic
   ```
   Imports C1.WPF
   Imports C1.WPF.Extended
   ```
   - C#
   ```
   using C1.WPF;
   using C1.WPF.Extended;
   ```
8. Add the following code just after the page's constructor to add handlers to the **Click** events:

   - Visual Basic
   ```
   Private Sub Button1_Click(ByVal sender as Object, ByVal e as
   System.Windows.RoutedEventArgs)
       Me.C1Book1.CurrentPage = Me.c1book1.CurrentPage - 1
   End Sub
   Private Sub Button2_Click(ByVal sender as Object, ByVal e as
   System.Windows.RoutedEventArgs)
       Me.C1Book1.CurrentPage = Me.c1book1.CurrentPage + 2
   End Sub
   ```
   - C#
   ```
   private void Button1_Click(object sender,
   System.Windows.RoutedEventArgs e)
   {
       this.c1Book1.CurrentPage = this.c1Book1.CurrentPage - 1;
   }
   private void Button2_Click(object sender,
   System.Windows.RoutedEventArgs e)
   {
       this.c1Book1.CurrentPage = this.c1Book1.CurrentPage + 1;
   }
   ```

   The buttons will now allow the user to navigate to the last or next page at run time.
9. Add code to the **Window_Loaded** event so that it appears similar to the following:

   - Visual Basic
   ```
   Private Sub Window1_Loaded(ByVal sender As System.Object, ByVal e As
   System.Windows.RoutedEventArgs) Handles MyBase.Loaded
       Dim txt1 As New TextBlock
       txt1.VerticalAlignment = VerticalAlignment.Center
       txt1.HorizontalAlignment = HorizontalAlignment.Center
       txt1.Text = "The End."
       C1Book1.Items.Add(txt1)
   End Sub
   ```
   - C#
   ```
   private void Window_Loaded(object sender, RoutedEventArgs e)
   {
       TextBlock txt1 = new TextBlock();
       txt1.VerticalAlignment = VerticalAlignment.Center;
       txt1.HorizontalAlignment = HorizontalAlignment.Center;
       txt1.Text = "The End.";
       c1Book1.Items.Add(txt1);
   ```

```
}
```

This will add a **TextBlock** to the C1Book control in code.

10. Save your project and return to XAML view.
11. In XAML view, add the following markup just after the `<Button x:Name="Button2"/>` tag:

- XAML to add

```
<Grid x:Name="checkers" Background="White" ShowGridLines="True">
    <Grid.RowDefinitions>
        <RowDefinition Height=".25*" />
        <RowDefinition Height=".25*" />
        <RowDefinition Height=".25*" />
        <RowDefinition Height=".25*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width=".25*" />
        <ColumnDefinition Width=".25*" />
        <ColumnDefinition Width=".25*" />
        <ColumnDefinition Width=".25*" />
    </Grid.ColumnDefinitions>
    <Rectangle Fill="Red" Grid.Row="0"  Grid.Column="0" Margin="5" />
    <Rectangle Fill="Black" Grid.Row="0" Grid.Column="1" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="0" Grid.Column="2" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="0" Grid.Column="3" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="1" Grid.Column="0" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="1" Grid.Column="1" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="1" Grid.Column="2" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="1" Grid.Column="3" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="2" Grid.Column="0" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="2" Grid.Column="1" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="2" Grid.Column="2" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="2" Grid.Column="3" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="3" Grid.Column="0" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="3" Grid.Column="1" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="3" Grid.Column="2" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="3" Grid.Column="3" Margin="5"  />
</Grid>
<Grid x:Name="checkers2" Background="White" ShowGridLines="True">
    <Grid.RowDefinitions>
        <RowDefinition Height=".25*" />
        <RowDefinition Height=".25*" />
        <RowDefinition Height=".25*" />
        <RowDefinition Height=".25*" />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width=".25*" />
        <ColumnDefinition Width=".25*" />
        <ColumnDefinition Width=".25*" />
        <ColumnDefinition Width=".25*" />
    </Grid.ColumnDefinitions>
    <Rectangle Fill="Red" Grid.Row="0" Grid.Column="0" Margin="5" />
    <Rectangle Fill="Black" Grid.Row="0" Grid.Column="1" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="0" Grid.Column="2" Margin="5"  />
   <Rectangle Fill="Black" Grid.Row="0" Grid.Column="3" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="1" Grid.Column="0" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="1" Grid.Column="1" Margin="5"  />
```

```
    <Rectangle Fill="Black" Grid.Row="1" Grid.Column="2" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="1" Grid.Column="3" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="2" Grid.Column="0" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="2" Grid.Column="1" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="2" Grid.Column="2" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="2" Grid.Column="3" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="3" Grid.Column="0" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="3" Grid.Column="1" Margin="5"  />
    <Rectangle Fill="Black" Grid.Row="3" Grid.Column="2" Margin="5"  />
    <Rectangle Fill="Red" Grid.Row="3" Grid.Column="3" Margin="5"  />
</Grid>
```

This markup will add two Grids with multiple Rectangle elements. This markup demonstrates how you can add multiple controls to a page of the C1Book control as long as the controls are all in one panel, such as a Grid or StackPanel.

In this step you completed adding content to the C1Book control. In the next step you'll run the application and observe run-time interactions.

Step 3 of 3: Running the Book Application

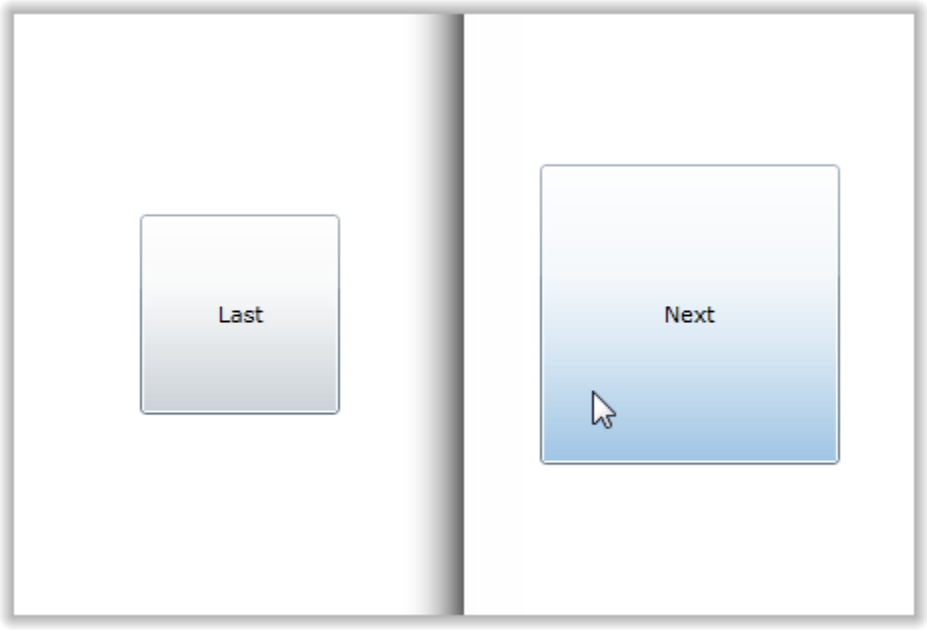
Now that you've created a WPF application and customized the application's appearance and behavior, the only thing left to do is run your application. To run your application and observe **Book for WPF**'s run-time behavior, complete the following steps:

1. From the **Debug** menu, select **Start Debugging** to view how your application will appear at run time.

   The application will appear similar to the following:

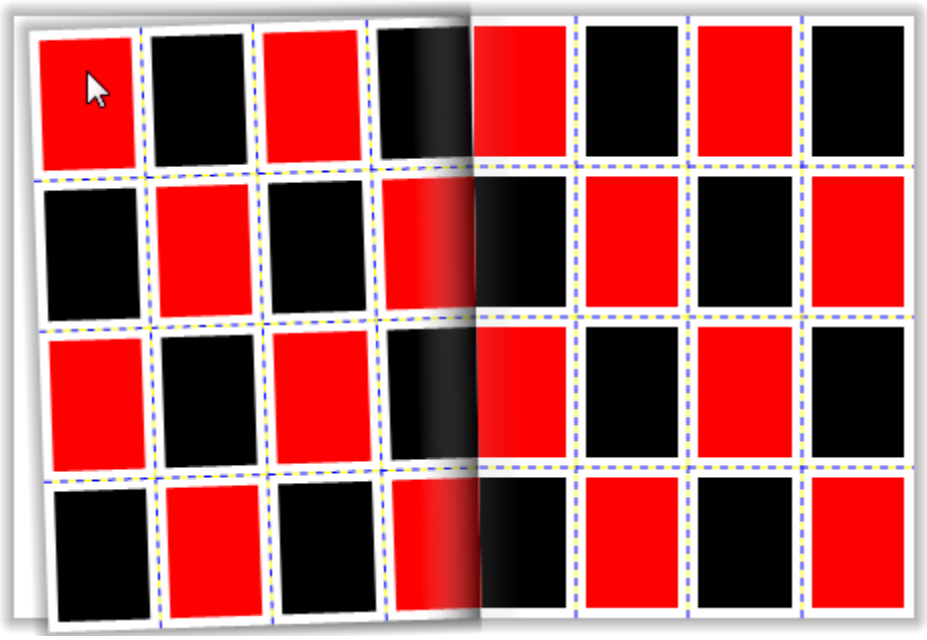
You set the IsFirstPageOnTheRight property so that only one page is initially visible. Notice that when you hover over the lower or upper-right corner of the C1Book control the page folds back slightly to prompt you to turn the page; see Book Zones (page 12) for more information.

2. Click the upper-right corner of the page and notice that the page turns and the second and third pages are visible:



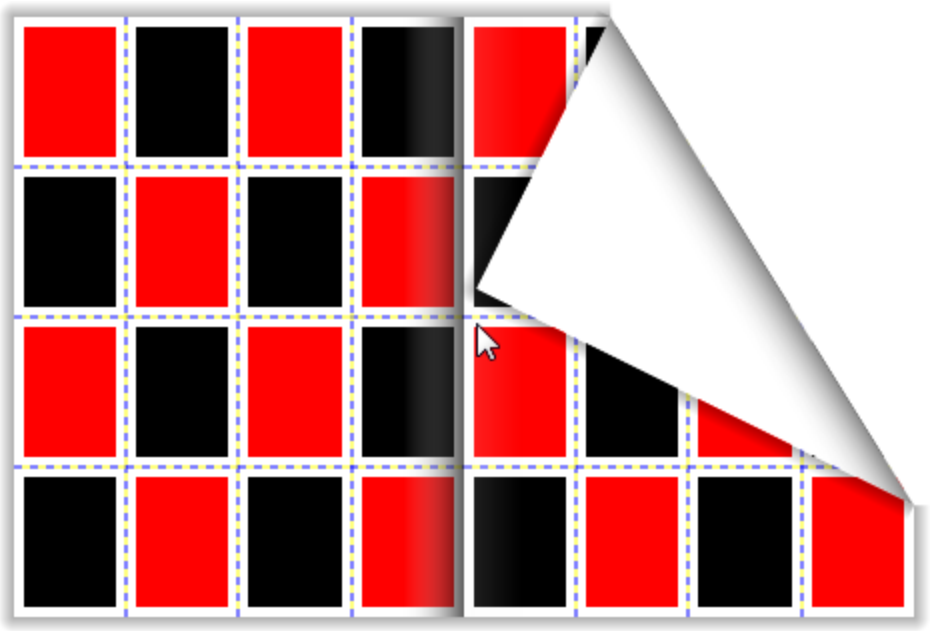
You customized the time it takes for the page to turn by setting the TurnInterval property.

3. Click the **Next** button. The next pages are displayed:

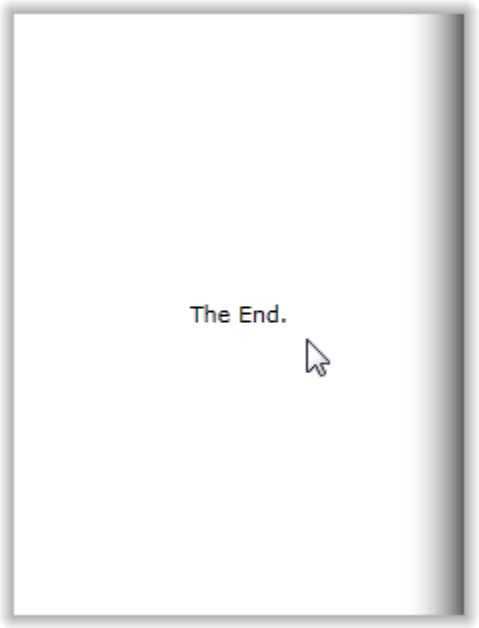


Note that you can return to the previous page by clicking the left top or bottom corner.

4. Click and drag the top-right page corner to the left to turn to the next page:



Notice that the last page contains the **TextBlock** content that was added in code:



Congratulations! You've completed the **Book for WPF** quick start and created a simple WPF application, added and customized a **Book for WPF** control, and viewed some of the run-time capabilities of the control.

Working with Book for WPF

ComponentOne Book for WPF includes the C1Book control, a simple book control that acts as a container, allowing you to add controls, images, and more in a familiar book format. When you add the C1Book control to a XAML window, it exists as a container, similar to a panel, that can be customized and include added content.

The control's interface looks similar to the following image:

XBAP Support

While most **Studio for WPF** products are fully functional in an XAML Browser Application (XBAP), **ComponentOne Book for WPF** does not include XBAP functionality in a partial trust environment. **Book for WPF** uses shader effects for the shadows in the book display and shader effects are unfortunately not allowed in a WPF partial trust environment.

To run an application as full trust, you can open the application's properties, and select **This is a full trust application** from the **Security** tab. For full instructions, see http://blogs.microsoft.co.il/blogs/maxim/archive/2008/03/05/wpf-xbap-as-full-trust-application.aspx.

Basic Properties

ComponentOne Book for WPF includes several properties that allow you to set the functionality of the control. Some of the more important properties are listed below. Note that you can see Book for WPF Appearance Properties (page 18) for more information about properties that control appearance.

The following properties let you customize the C1Book control:

Property	Description
CurrentPage	Gets or sets the current page that is displayed.
CurrentZone	Gets the zone of the book under the mouse. See Book Zones (page 12) for more information.
FoldSize	Gets or sets the size of the fold (in pixels). See Page Fold

	Size (page 14) for more information.
IsFirstPageOnTheRight	Gets or sets whether the first page of the book is displayed in the right side of the book. See First Page Display (page 16) for more information.
LeftPageTemplate	Gets or sets the template of the page shown in the left side of the book.
PageFoldAction	Gets or sets the action that will raise the turn animation. See Page Turning Options (page 15) for more information.
RightPageTemplate	Gets or sets the template of the page shown in the right side of the book.
ShowInnerShadows	Gets or sets whether the inner shadows are shown. See Page Shadows (page 17) for more information.
ShowOuterShadows	Gets or sets whether the outer shadows are shown. See Page Shadows (page 17) for more information.
ShowPageFold	Gets or sets whether the fold is displayed. See Page Fold Visibility (page 15) for more information.
TurnInterval	Gets or sets the amount (in milliseconds) of time of the turn animation.

Basic Events

ComponentOne Book for WPF includes several events that allow you to set interaction and customize the control. Some of the more important events are listed below.

The following events let you customize the C1Book control:

Event	Description
CurrentPageChanged	Fires when the CurrentPage property changes.
CurrentZoneChanged	Occurs when current zone changed. For more information, see Book Zones (page 12).
DragPageFinished	Occurs when page ends of being dragged.
DragPageStarted	Occurs when the page starts to be dragged.
IsMouseOverChanged	Event raised when the **IsMouseOver** property has changed.

Book Zones

The C1Book control includes several zones. These zones let you customize what happens when users interact with various sections of the control. You can use the CurrentZone property to get the user's current zone and you can use the CurrentZoneChanged event to customize when happens when users move to a different zone.

There are six separate zones in the C1Book control. For an illustration of each zone, note the mouse's position in each of the images in the following table:

Zone	Description	Example

Out	Specifies the zone outside the borders of the book.	
BottomLeft	Specifies bottom left fold zone.	
TopLeft	Specifies top left fold zone.	

Center	Specifies the center of the book (no fold zone).	
TopRight	Specifies top right fold zone.	
BottomRight	Specifies bottom left fold zone.	

Page Fold Size

One way to customize the appearance of the book as users flip pages is by setting the size of the page fold using the FoldSize property. Page folds, which appear when users mouse over certain book zones (page 12), serve as a cue to users that a page can be turned.

When you set the FoldSize property you will be setting the size of all of the page folds – this includes the right top and bottom folds and the left top and bottom folds. So for example, when FoldSize is **40**, the bottom left and bottom right folds appear similar to the following image:

If set to a higher number, the folds will appear more prominent. When FoldSize is **80**, the bottom left and bottom right folds appear similar to the following image:

Page Fold Visibility

By default, users will see a page fold when their mouse is over certain book zones (page 12). If you choose to, however, you can change the page fold visibility. You can set the ShowPageFold property to any of the values in the following table to determine how users interact with the C1Book control:

Value	Description
OnMouseOver	The fold will be visible when the user drags the mouse over the edge of the page. This is the default setting.
Never	The fold will not be visible.
Always	The fold will always be visible.

Page Turning Options

By default, when users click once on a page fold the book will progress to the previous or next page. You can customize how pages turn on page click using the PageFoldAction property. For example you can set

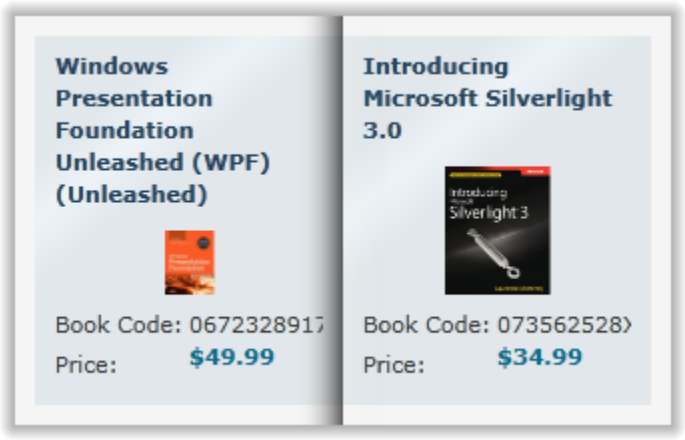
PageFoldAction so that users must double-click on the page fold to turn the page, or you can prevent page turning on mouse click altogether, requiring that uses perform a drag-and-drop operation on the page fold to turn a page.

You can set the PageFoldAction property to any of the values in the following table to determine how users interact with the C1Book control:

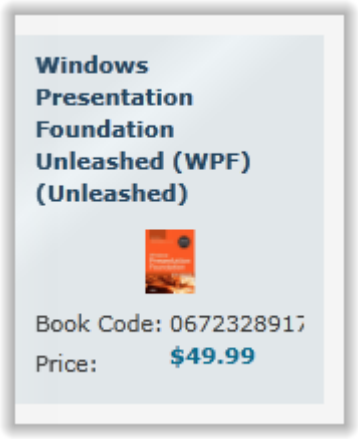
Value	Description
TurnPageOnClick	Turn the page when the user clicks the page fold.
TurnPageOnDoubleClick	Turn the page when the user double clicks the page fold.
None	Turn page when user drags the page fold across the book.

First Page Display

By default, the first page in the C1Book control is displayed on the left hand side. This makes it appear as if the book is open:



If you choose, however, you can change the display of the first page to appear on the right by setting the IsFirstPageOnTheRight property to **True**. When the first page is set to display on the right side, it will appear similar to a cover, as if the book is closed:



Page Shadows

The C1Book control includes shadows that give the control a three-dimensional appearance. The control includes inner and outer shadows. Inner shadows appear in the center of the book, and behind the right page when turning. Outer shadows appear around the outside of the control and behind the left page when turning. For example, the following image illustrates inner and outer shadows:

If you do not want shadows to be displayed, you can change their visibility by setting the ShowInnerShadows and ShowOuterShadows properties to **False**.

Book Navigation

At run time users can navigate through the C1Book control using the mouse. Users can click in one of the book zones (page 12) or can perform a drag-and-drop operation to turn the page. The C1Book control includes navigation-related methods, properties, and events to make it easier for you to determine what page a user is currently viewing, and to set the application's actions as users navigate through a book.

The CurrentPage property gets or sets the page that is currently displayed at run time. Note that when you turn a page, the page displayed on the left of the two-page spread will be the CurrentPage. Page numbering begins with **0**

and page 0 is always displayed on the left. So if IsFirstPageOnTheRight property is set to **True**, the first initial page of the book displayed on the right side will be page 1 with a hidden page 0 on the left side.

You can set the displayed page using the CurrentPage property, but you can also use the TurnPage method to change the current page at run time. The TurnPage method turns to book pages forward or back one page.

You can use the CurrentPageChanged event to specify actions that happen when the current page is changed. You can also use the DragPageStarted and DragPageFinished events to specify actions to take when the user turns the page using a drag-and-drop operation.

Book for WPF Layout and Appearance

The following topics detail how to customize the C1Book control's layout and appearance. You can use built-in layout options to lay your controls out in panels such as Grids or Canvases. Themes allow you to customize the appearance of the grid and take advantage of WPF's XAML-based styling. You can also use templates to format and lay out the control and to customize the control's actions.

Book for WPF Appearance Properties

ComponentOne Book for WPF includes several properties that allow you to customize the appearance of the control. You can change the color, border, and height of the control. The following topics describe some of these appearance properties.

Color Properties

The following properties let you customize the colors used in the control itself:

Property	Description
Background	Gets or sets a brush that describes the background of a control. This is a dependency property.
Foreground	Gets or sets a brush that describes the foreground color. This is a dependency property.

Alignment Properties

The following properties let you customize the control's alignment:

Property	Description
HorizontalAlignment	Gets or sets the horizontal alignment characteristics applied to this element when it is composed within a parent element, such as a panel or items control. This is a dependency property.
VerticalAlignment	Gets or sets the vertical alignment characteristics applied to this element when it is composed within a parent element such as a panel or items control. This is a dependency property.

Border Properties

The following properties let you customize the control's border:

Property	Description

BorderBrush	Gets or sets a brush that describes the border background of a control. This is a dependency property.
BorderThickness	Gets or sets the border thickness of a control. This is a dependency property.

Size Properties

The following properties let you customize the size of the control:

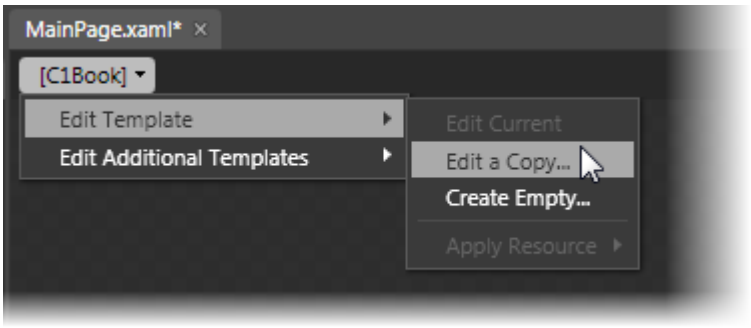
Property	Description
Height	Gets or sets the suggested height of the element. This is a dependency property.
MaxHeight	Gets or sets the maximum height constraint of the element. This is a dependency property.
MaxWidth	Gets or sets the maximum width constraint of the element. This is a dependency property.
MinHeight	Gets or sets the minimum height constraint of the element. This is a dependency property.
MinWidth	Gets or sets the minimum width constraint of the element. This is a dependency property.
Width	Gets or sets the width of the element. This is a dependency property.

Book Templates

One of the main advantages to using a WPF control is that controls are "lookless" with a fully customizable user interface. Just as you design your own user interface (UI), or look and feel, for WPF applications, you can provide your own UI for data managed by **ComponentOne Book for WPF**. Extensible Application Markup Language (XAML; pronounced "Zammel"), an XML-based declarative language, offers a simple approach to designing your UI without having to write code.

Accessing Templates

You can access templates in Microsoft Expression Blend by selecting the C1Book control and, in the menu, selecting **Edit Control Parts (Templates)**. Select **Edit a Copy** to create an editable copy of the current template or **Create Empty**, to create a new blank template.



Note: If you create a new template through the menu, the template will automatically be linked to that template's property. If you manually create a template in XAML you will have to link the appropriate template property to the template you've created.

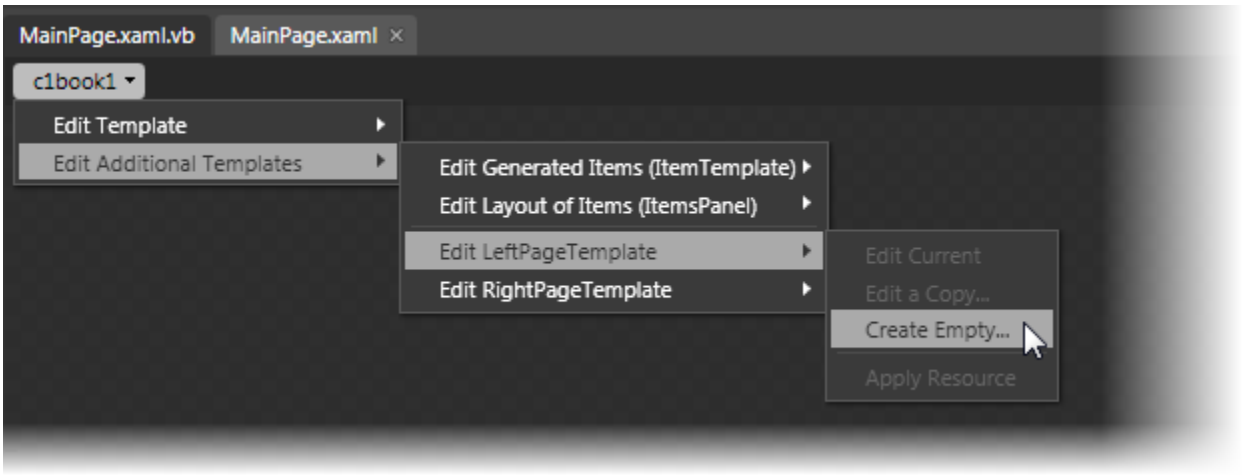
Note that you can use the Template property to customize the template.

Page Templates

ComponentOne Book for WPF's book control includes two page templates: the LeftPageTemplate and the RightPageTemplate. These templates control the appearance and layout of the left and right pages of the book. These templates function as master pages -- items that you add to these page templates will appear on every page that template effects. This is useful, for example, if you want to add a watermark or company logo to every page without adding it to every single page in the book individually.

Accessing Page Templates

You can access page templates in Microsoft Expression Blend by selecting the C1Book control and, in the menu, selecting **Edit Additional Templates**. Choose **Edit LeftPageTemplate** or **Edit RightPageTemplate** and select **Create Empty** to create a new blank template.



The **Create ControlTemplate Resource** dialog box will appear allowing you to name the template and determine where to define the template. By default, the template will appear blank with an empty Grid control:

```
<ControlTemplate x:Key="NewLeftPageTemplate">
    <Grid/>
</ControlTemplate>
```

You can customize the template as you would any other **ControlTemplate**.

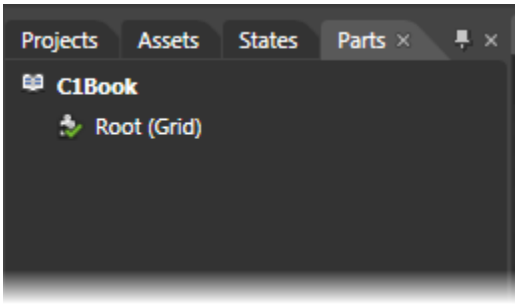
Book Styles

ComponentOne Book for WPF's C1Book control provides several style properties that you can use to change the appearance of the control. Some of the included styles are described in the table below:

Style	Description
FontStyle	Gets or sets the font style. This is a dependency property.
Style	Gets or sets the style used by this element when it is rendered. This is a dependency property.

Book Template Parts

In Microsoft Expression Blend, you can view and edit template parts by creating a new template (for example, click the C1Book control to select it and choose **Object | Edit Template | Edit a Copy**). Once you've created a new template, the parts of the template will appear in the **Parts** window:



Note that you may have to select the **ControlTemplate** for its parts to be visible in the **Parts** window.

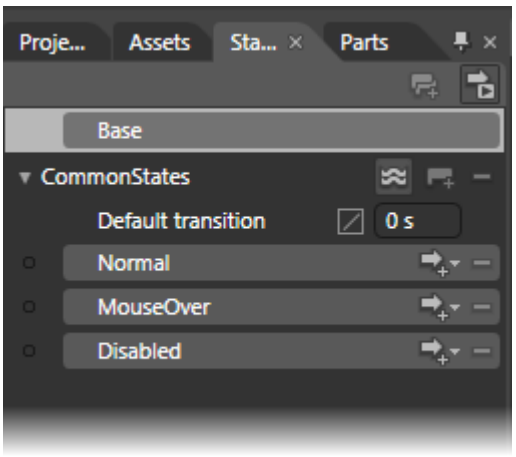
In the Parts window, you can double-click any element to create that part in the template. Once you have done so, the part will appear in the template and the element's icon in the **Parts** pane will change to indicate selection.

Template parts available in the C1Book control include:

Name	Type	Description
Root	FrameworkElement	Provides a framework of common APIs for objects that participate in WPF layout. Also defines APIs related to data binding, object tree, and object lifetime feature areas in WPF.

Book Visual States

In Microsoft Expression Blend, you can add custom states and state groups to define a different appearance for each state of your user control – for example, the visual state of the control could change on mouse over. You can view and edit visual states by creating a new template and adding a new template part (page 21). Once you've done so the available visual states for that part will be visible in the **Visual States** window:



Common states include **Normal** for the normal appearance of the item, **MouseOver** for the item on mouse over, and **Disabled** for when the item is not enabled. Focus states include **Unfocused** for when the item is not in focus and **Focused** when the item is in focus.

Book for WPF Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos, which may make use of other ComponentOne development tools included with the ComponentOne Studios. Samples can be accessed from the **ComponentOne Studio for WPF ControlExplorer**. To view samples, on your desktop, click the **Start** button and then click **All Programs | ComponentOne | Studio for WPF | Samples | WPF ControlExplorer**.

C# Samples

The following C# sample is included:

Sample	Description
ControlExplorer	The **Book** page in the **ControlExplorer** sample demonstrates how to add content to and customize the C1Book control.

Book for WPF Task-Based Help

The following task-based help topics assume that you are familiar with Visual Studio and Expression Blend and know how to use the C1Book control in general. If you are unfamiliar with the **ComponentOne Book for WPF** product, please see the Book for WPF Quick Start (page 3) first.

Each topic in this section provides a solution for specific tasks using the **ComponentOne Book for WPF** product. Most task-based help topics also assume that you have created a new WPF project and added a C1Book control to the project – for information about creating the control, see Creating a Book (page 23).

Creating a Book

You can easily create a C1Book control at design time, in XAML, and in code. Note that if you create a C1Book control as in the following steps, it will appear as an empty container. You will need to add items to the control for it to appear as a book at run time. For an example, see Adding Items to a Book (page 25).

At Design Time

To create a C1Book control, complete the following steps:

1. Click once on the design surface of the Window to select it.
2. Double-click the **C1Book** icon in the Toolbox to add the control to the panel. The C1Book control will now exist in your application.
3. If you choose, you can customize the control by selecting it and setting properties in the Properties window.

In XAML

To create a C1Book control using XAML markup, complete the following steps:

1. In the Visual Studio Solution Explorer, right-click the **References** folder in the project files list. In the context menu choose **Add Reference**, select the **C1.WPF.dll** and **C1.WPF.Extended.dll** assemblies, and click **OK**.
2. Add a XAML namespace to your project by adding `xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"` to the initial `<Window>` tag. It will appear similar to the following:

```
<Window
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
x:Class="Window1" Title="Window1" Height="230" Width="348">
```

3. Add a `<c1:C1Book>` tag to your project within the `<Grid>` tag to create a C1Book control. The markup will appear similar to the following:

```
<Grid>
    <c1:C1Book x:Name="C1Book1" Height="300" Width="450"/>
    </c1:C1Book>
</Grid>
```

This markup will create an empty C1Book control named "C1Book1" and set the control's size.

In Code

To create a C1Book control in code, complete the following steps:

1. In the Visual Studio Solution Explorer, right-click the **References** folder in the project files list. In the context menu choose **Add Reference**, select the **C1.WPF.dll** and **C1.WPF.Extended.dll** assemblies, and click **OK**.
2. In XAML view, give the initial grid in the window a name, by updating the tag so it appears similar to the following:

```
<Grid x:Name="LayoutRoot">
```

3. Right-click within the **Window1.xaml** window and select **View Code** to switch to Code view.
4. Add the following import statements to the top of the page:
   - Visual Basic

```
Imports C1.WPF
Imports C1.WPF.Extended
```

   - C#

```
using C1.WPF;
using C1.WPF.Extended;
```

5. Add code to the page's constructor to create the C1Book control. It will look similar to the following:
   - Visual Basic

```
Public Sub New()
    InitializeComponent()
    Dim c1book1 as New C1Book
    c1book1.Height = 300
    c1book1.Width = 450
    LayoutRoot.Children.Add(c1book1)
End Sub
```

   - C#

```
public MainPage()
{
    InitializeComponent();
    C1Book c1book1 = new C1Book();
    c1book1.Height = 300;
    c1book1.Width = 450;
    LayoutRoot.Children.Add(c1book1);
}
```

This code will create an empty C1Book control named "c1book1", set the control's size, and add the control to the page.

What You've Accomplished

You've created a C1Book control. Note that when you create a C1Book control as in the above steps, it will appear as an empty container. You will need to add items to the control for it to appear as a book at run time. For an example, see Adding Items to a Book (page 25).

Adding Items to a Book

You can add any sort of arbitrary content to a C1Book control. This includes text, images, layout panels, and other standard and 3rd-party controls. In this example, you'll add a **TextBlock** control to a C1Book control, but you can customize the steps to add other types of content instead.

At Design Time

To add a **TextBlock** control to the book, complete the following steps:

1. Click the C1Book control once to select it.
2. Navigate to the Toolbox, and double-click the **TextBlock** item to add the control to the C1Book control.
3. If you choose, you can customize the C1Book and **TextBlock** controls by selecting each control and setting properties in the Properties window. For example, set the **TextBlock**'s **Text** property to "Hello World!".

In XAML

For example, to add a **TextBlock** control to the book add `<TextBlock Text="Hello World!"/>` within the `<c1:C1Book>` tag so that it appears similar to the following:

```
<c1:C1Book x:Name="C1Book1" Height="300" Width="450">
    <TextBlock Text="Hello World!"/>
</c1:C1Book>
```

In Code

For example, to add a **TextBlock** control to the book, add code to the page's constructor so it appears like the following:

- Visual Basic

```
Public Sub New()
    InitializeComponent()
    Dim txt1 as New TextBlock
    txt1.Text = "Hello World!"
    C1Book1.Items.Add(txt1)
End Sub
```

- C#

```
public MainPage()
{
    InitializeComponent();
    TextBlock txt1 = new TextBlock();
    txt1.Text = "Hello World!";
    c1Book1.Items.Add(txt1);
}
```

What You've Accomplished

You've added a control to the C1Book control. Run the application and observe that the **TextBlock** control has been added to the C1Book control. You can similarly add other content and controls.

Clearing Items in a Book

You may choose to allow users to clear all items from the C1Book control at run time, or you may need to clear the items collection when binding and then rebinding the control to another data source.

For example, to clear the book's content add the following code to your project:

- Visual Basic
```
Me.C1Book1.Items.Clear()
```
- C#
```
this.c1Book1.Items.Clear();
```

What You've Accomplished

The control's content will be cleared. If you run the application, you will observe that the book is blank.

Displaying the First Page on the Right

The IsFirstPageOnTheRight property gets or sets if the first page of the book is displayed on the right or the left side. See First Page Display (page 16) for more information. By default the C1Book control starts with the first page displayed on the left and two pages displayed, but you can customize this by setting the IsFirstPageOnTheRight property at design time, in XAML, and in code.

At Design Time

To set the IsFirstPageOnTheRight property at design time, complete the following steps:

1. Click the C1Book control once to select it.
2. Navigate to the Properties window and check the check box next to the **IsFirstPageOnTheRight** item.

In XAML

For example, to set the IsFirstPageOnTheRight property, add `IsFirstPageOnTheRight="True"` to the `<c1:C1Book>` tag so that it appears similar to the following:

```
<c1:C1Book x:Name="C1Book1" Height="300" Width="450"
IsFirstPageOnTheRight="True">
```

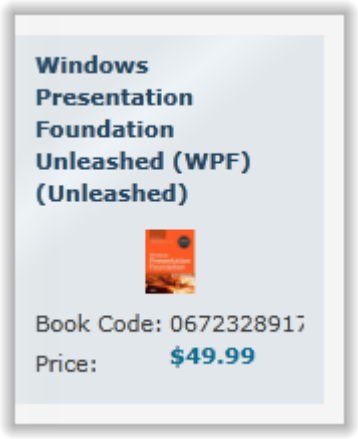
In Code

For example, to set the IsFirstPageOnTheRight property, add the following code to your project in the page's constructor:

- Visual Basic
```
Me.C1Book1.IsFirstPageOnTheRight = True
```
- C#
```
this.c1Book1.IsFirstPageOnTheRight = true;
```

What You've Accomplished

You've set the first page to appear on the right. If you run the application, the first page will appear as a single page, like the book's cover:



Setting the Initial Page

The CurrentPage property gets or sets the value of the C1Book control's current page. By default the C1Book control starts with the first page displayed but you can customize this by setting the CurrentPage property at design time, in XAML, and in code.

At Design Time

To set the CurrentPage property to **3** at design time, complete the following steps:

1. Click the C1Book control once to select it.
2. Navigate to the Properties window and click in the text box next to the **CurrentPage** item.
3. Enter a number, for example "3", for the displayed initial page.

In XAML

For example, to set the CurrentPage property to **3**, add `CurrentPage="3"` to the `<c1:C1Book>` tag so that it appears similar to the following:

```
<c1:C1Book x:Name="C1Book1" Height="300" Width="450" CurrentPage="3">
```

In Code

For example, to set the CurrentPage property to **3**, add the following code to your project:

- Visual Basic

```
Me.C1Book1.CurrentPage = 3
```

- C#

```
this.c1Book1.CurrentPage = 3;
```

What You've Accomplished

You've changed the book's initial starting page. If you run the application, the initial page that appears will be page 3.

Navigating the Book with Code

You can set the displayed page using the CurrentPage property, but you can also use the TurnPage method to change the current page at run time. For more information, see Book Navigation (page 17). In this topic you'll add two buttons to your application, one that will turn to the previous page and one that will turn to the next page of the book.

To add additional navigation to your book, complete the following steps:

1. Navigate to the Toolbox and double-click the **Button** item twice to add two **Button** controls to your application.
2. Select **Button1**, navigate to the Properties window and set the **Content** property to "<".
3. Select **Button2**, navigate to the Properties window and set the **Content** property to ">".
4. Resize and reposition the buttons in the Window. Place the **Button1** button to the left of the book, and the **Button2** button to the right of the book.
5. Double-click **Button1** to create the **Button_Click** event handler and switch to Code view.
6. Return to Design view and repeat the previous step with **Button2** so each button has a **Click** event specified.

   The XAML markup will appear similar to the following:

```
<Button HorizontalAlignment="Right" Margin="0,43,12,0" Name="Button1"
Width="28" Height="23" VerticalAlignment="Top">&gt;</Button>
<Button Height="23" HorizontalAlignment="Left" Margin="12,43,0,0"
Name="Button2" VerticalAlignment="Top" Width="28">&lt;</Button>
```

7. Switch to Code view and add the following import statements to the top of the page:

   - Visual Basic

```
Imports C1.WPF
Imports C1.WPF.Extended
```

   - C#

```
using C1.WPF;
using C1.WPF.Extended;
```

8. Add code to the **Click** event handlers so they look like the following:

   - Visual Basic

```
Private Sub Button1_Click(ByVal sender as Object, ByVal e as
System.Windows.RoutedEventArgs)
    Me.C1Book1.TurnPage(True)
End Sub

Private Sub Button2_Click(ByVal sender as Object, ByVal e as
System.Windows.RoutedEventArgs)
    Me.C1Book1.TurnPage(False)
End Sub
```

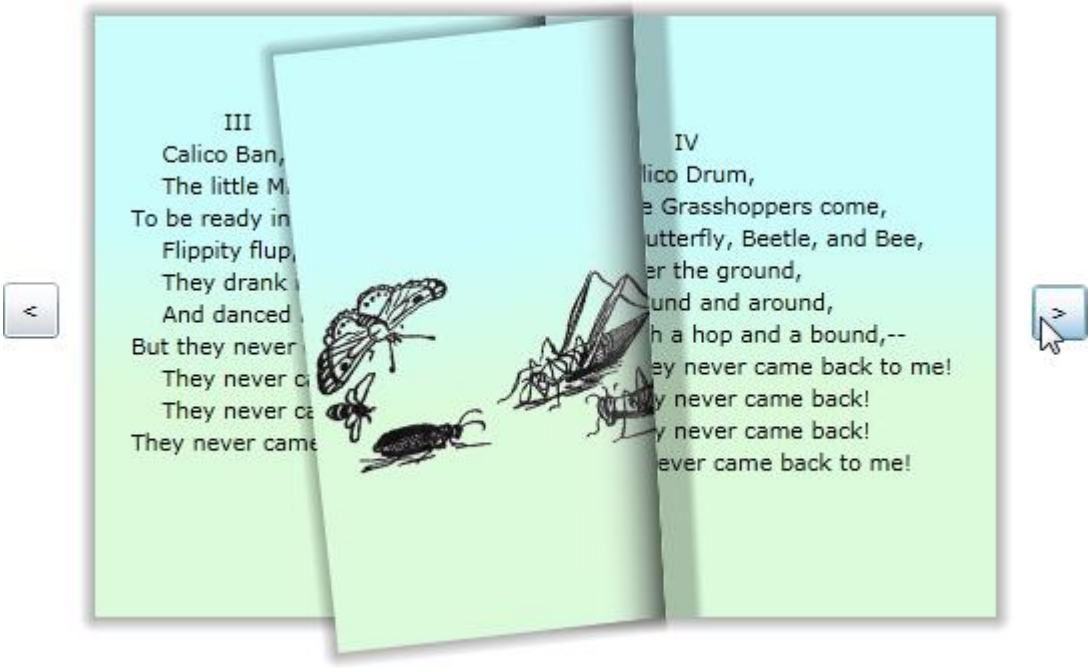
   - C#

```
public MainPage()
{
private void button1_Click(object sender,
System.Windows.RoutedEventArgs e)
{
    this.c1Book1.TurnPage(true);
}
private void button2_Click(object sender,
System.Windows.RoutedEventArgs e)
{
    this.c1Book1.TurnPage(false);
}}
```

   This code will turn the book a page forward or back depending on the button clicked.

What You've Accomplished

You've customized navigation in the book. To view the book's navigation, run the application and click the right button. Notice that the page turns to the next page with a page turning animation:



Click the left button and observe that the book returns to the previous page.