

Extended Library for UWP

2018.04.10 更新

グレースィティ株式会社

目次

Extended Library for UWP	3
Book for UWP	3
主な特長	3
XAML クイックリファレンス	3
クイックスタート	4
手順1:Book アプリケーションの作成	4
手順2:Book コントロールへのコンテンツの追加	4-8
手順3:アプリケーションへのファイルの追加	8-10
手順4:Book アプリケーションの実行	10-11
C1Book の使い方	11-12
基本的なプロパティ	12
基本的なイベント	12-13
ブックのゾーン	13-15
ページの折り返しのサイズ	15-16
ページの折り返しの表示/非表示	16-17
ページめくりオプション	17
最初のページの表示	17-18
ブックナビゲーション	18
レイアウトおよび外観	18
色のプロパティ	18-19
境界線のプロパティ	19
サイズのプロパティ	19
ブックのスタイル	19
タスク別ヘルプ	19
ブックの作成	19-21
ブックへの項目の追加	21-22
ブック内の項目のクリア	22
最初のページを右側に表示	22-23
現在のページの設定	23-24
コードでのブック内のナビゲーション	24-30
ColorPicker for UWP	30

主な特長	30-32
ビジュアル要素	32-35
クイックリファレンス	35
クイックスタート	35-36
手順1:C1ColorPicker アプリケーションの作成	36-37
手順2:C1ColorPicker コントロールの追加	37
手順3:アプリケーションにコードの追加	37-39
手順4:アプリケーションの実行	39-42
C1ColorPicker の使い方	42-43
パレットの設定	43-44
パレットのカスタマイズ	44-46
背景色の設定	46-47
ドロップダウン方向の変更	47-48
基本モードから最近使用した色の非表示	48-49

Extended Library for UWP

Extended Library には、以下のコントロールが含まれます。

- **Book for UWP**

Book for UWPは、**UIElement**オブジェクトを実際の本のページと同様に表示します。タッチ操作でスムーズなページめくりアニメーションを行うことができ、インタラクティブでユニークなデータ視覚化方法を提供します。

- **ColorPicker for UWP**

ColorPicker for UWPを使用して、プロフェッショナルなパレットから色を選択したり、透過性を取り込んだカスタム色を作成したりすることができます。

Book for UWP

Book for UWP は、本のようにページをめくることができる革新的なナビゲーションコントロールです。この見慣れた本の体裁を使用して、情報を表示します。**Book for UWP** は、**UIElement** オブジェクトを普通の本のページのように表現できます。**Book for UWP** を使用すると、一度に2つの要素を表示したり、ジェスチャによってページをめくることがなどできます。

主な特長

Book for UWP を使用すると、機能豊富でカスタマイズされたアプリケーションを作成できます。**Book for UWP** は、次の主な特長を備えています。

- **見慣れた本の体裁**

C1Book は、本という見慣れたモデルを使用して、情報を革新的に表現することができます。ただし、**Book for UWP** は通常の動かない本ではなく、ダイナミックでインタラクティブです。

- **本物の本のような画像**

C1Book では、ページの外観をカスタマイズできます。たとえば、ページの折り返しを表示することができます。本のように見えるだけでなく、本のように扱うことができます。

- **柔軟なデータ連結**

C1Book は **ItemsControl** なので、どのデータソースにも連結可能です。データソースの各項目は、**UIElement** にするか、テンプレートを使用して **UIElement** に変換される汎用オブジェクトにすることができます。

- **ジェスチャベースのナビゲーション**

- **C1Book** を使用すると、ジェスチャによってページ間を移動することができます。タップ、スワイプ、スライドのジェスチャを使用して、ページをめくることができます。

XAML クイックリファレンス

このトピックでは、5つの項目を含む **C1Book** コントロールを作成し、現在のページを3(0から始まるインデックスなので4ページ目)に設定するための XAML の概要を提供します。詳細については、「[タスク別ヘルプ](#)」を参照してください。

XAML マークアップ

```
<Extended:C1Book x:Name="c1book1" Height="300" Width="450" CurrentPage="3">
<TextBlock Text="Hello World! 1"/>
<TextBlock Text="Hello World! 2"/>
<TextBlock Text="Hello World! 3"/>
<TextBlock Text="Hello World! 4"/>
<TextBlock Text="Hello World! 5"/>
</Extended:C1Book>
```

クイックスタート

このクイックスタートガイドは、**Book for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートでは、新しいプロジェクトを作成し、アプリケーションに **C1Book** コントロールを追加して、コントロールの外観と動作をカスタマイズします。

手順1: Book アプリケーションの作成

この手順では、Visual Studio で新しい UWP スタイルのアプリケーションを作成します。C1Book コントロールをアプリケーションに追加すると、完全に機能する本のようなインタフェースになり、そこに画像、コントロールなどの要素を追加することができます。プロジェクトをセットアップし、**C1Book** コントロールをアプリケーションに追加するには、次の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新規作成]**→**[プロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで、左ペインの言語を展開します。
 - a. 言語の下で、**[Windows ストア]**を選択します。
 - b. テンプレートリストで、**[新しいアプリケーション (XAML)]**を選択します。
 - c. **名前**を入力し、**[OK]**をクリックしてプロジェクトを作成します。
3. プロジェクトが開いたら、**[プロジェクト]**ウィンドウに移動し、プロジェクトファイルリストで**[参照]**フォルダを右クリックします。コンテキストメニューで、**[参照の追加]**を選択します。
4. **[参照マネージャ]**で、**ComponentOne for UWP SDK**を選択します。
5. MainPage.xaml が開いていない場合は開きます。`<Grid>` タグと `</Grid>` タグの間にカーソルを置き、1回クリックします。
6. ツールボックスに移動し、C1Book アイコンをダブルクリックして、コントロールをグリッドに追加します。これで、参照と XAML 名前空間が自動的に追加されます。XAML マークアップは次のようになります。

マークアップ

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
    <Extended:C1Book />
</Grid>
```

7. デザインビューで C1Book コントロールをクリックし、**[プロパティ]**ウィンドウに移動して、次のプロパティを設定します。
 - コード内でコントロールにアクセスできるように、**Name** を "book" に設定して名前を付けます。
 - **[Width]**を "450" に、**[Height]**を "300" に設定します。
 - **IsFirstPageOnTheRight** を "true" に設定します。

XAML は次のようになります。

マークアップ

```
<Grid>
    <Extended:C1Book x:Name="book" Height="450" Width="600">
    </Extended:C1Book>
</Grid>
```

これで、アプリケーションのユーザーインタフェースのセットアップは終了しましたが、C1Book コントロールにはまだ何もコンテンツが入っていません。次の手順では、C1Book コントロールにコンテンツを追加し、さらにアプリケーションにコードを追加して、コントロールに機能を追加します。

手順2: Book コントロールへのコンテンツの追加

この手順では、設計時、XAML マークアップ、およびコードで **C1Book** コントロールにコンテンツを追加して、ページめくり可能な複数のページから成る仮想の本を作成します。プロジェクトをカスタマイズしてアプリケーションの C1Book コントロールにコ

コンテンツを追加するには、次の手順に従います。

1. 次のようにマークアップを編集します。

マークアップ

```
<Border Grid.Row="1">
  <Grid>
    <Extended:C1Book x:Name="book" Height="450" Width="600">
      </Extended:C1Book>
    </Grid>
  </Border>
```

2. `<Extended: C1Book>` `</Extended: C1Book>` タグ内で、次の XAML マークアップを追加します。これで、マークアップにいくつかのテンプレートが追加されます。

▶ XAML でマークアップの書き方

マークアップ

```
<Extended:C1Book.LeftPageTemplate>
  <ControlTemplate TargetType="ContentControl">
    <Border Background="WhiteSmoke" BorderBrush="WhiteSmoke"
BorderThickness="10 10 0 10">
      <ContentPresenter Content="{TemplateBinding
Content}" ContentTemplate="{TemplateBinding ContentTemplate}" Margin="{
TemplateBinding Padding}" />
    </Border>
  </ControlTemplate>
</Extended:C1Book.LeftPageTemplate>
<Extended:C1Book.RightPageTemplate>
  <ControlTemplate TargetType="ContentControl">
    <Border Background="WhiteSmoke" BorderBrush="WhiteSmoke"
BorderThickness="10 10 0 10">
      <ContentPresenter Content="{TemplateBinding
Content}" ContentTemplate="{TemplateBinding ContentTemplate}" Margin="{
TemplateBinding Padding}" />
    </Border>
  </ControlTemplate>
</Extended:C1Book.RightPageTemplate>
<Extended:C1Book.ItemTemplate>
  <DataTemplate>
    <Grid>
      <Grid.Background>
        <LinearGradientBrush EndPoint="1,1"
StartPoint="0,0">
          <GradientStop Color="#FFE2E8EB"
Offset="0.2"/>
          <GradientStop Color="#FFEEF4F7"
Offset="0.3"/>
          <GradientStop Color="#FFE2E8EB"
Offset="0.4"/>
        </LinearGradientBrush>
      </Grid.Background>
      <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
      </Grid.RowDefinitions>
```

```

        <RowDefinition Height="*" />
        <RowDefinition Height="Auto" />
    </Grid.RowDefinitions>
    <Image Source="{Binding Path=CoverUri}"
Stretch="Uniform" HorizontalAlignment="Center" VerticalAlignment="Center"
Grid.Row="1" />
        <TextBlock Text="{Binding Path=Title}"
TextWrapping="Wrap" TextAlignment="Left" FontSize="11" FontWeight="Bold"
Margin="10,7,10,10" Foreground="#FF22445F" />
        <Grid HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" Grid.Row="2" Grid.RowSpan="1" Margin="10,7,10,10">
            <Grid.ColumnDefinitions>
                <ColumnDefinition />
                <ColumnDefinition />
            </Grid.ColumnDefinitions>
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>
            <TextBlock HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" Text="{Binding Path=Price}" TextWrapping="NoWrap"
Grid.ColumnSpan="1" Grid.Row="1" Grid.Column="1" FontSize="11"
Foreground="#FF086C8E" FontWeight="Bold" />
            <TextBlock HorizontalAlignment="Stretch"
VerticalAlignment="Stretch" Text="{Binding Path=Id}" Grid.ColumnSpan="1"
Grid.Column="1" TextWrapping="NoWrap" FontSize="11" Foreground="#FF383838" />
            <TextBlock Text="Book Code:"
TextWrapping="NoWrap" FontSize="11" Foreground="#FF383838" />
            <TextBlock Text="Price:" TextWrapping="NoWrap"
Grid.Row="1" Margin="0,4,0,2" FontSize="11" Foreground="#FF383838" />
        </Grid>
    </Grid>
</DataTemplate>
</Extended:C1Book.ItemTemplate>

```

3. コードビューで、次の import 文をページの先頭に追加します。

▶ C# コードの書き方

```

C#
using C1.Xaml.Extended;

```

4. ページのコンストラクタの直後に次のコードを追加します。このコードで、別のコードファイルからデータを呼び出すことができます。

▶ C# コードの書き方

```

C#
InitDataSource();
}
private void InitDataSource()
{
    // xml から本の説明をロードします
    string peopleXMLPath =

```

```
Path.Combine(Package.Current.InstalledLocation.Path, "Amazon.xml");
XDocument doc = XDocument.Load(peopleXMLPath);
// XDocument doc = XDocument.Load(@"..\..\..\Amazon.xml");
var books = from reader in doc.Descendants("book")
            select new AmazonBookDescription
            {
                Title = reader.Attribute("title").Value,
                CoverUri = reader.Attribute("coverUri").Value,
                Id = reader.Attribute("id").Value,
                Price = reader.Attribute("price").Value,
                StockAmount =
int.Parse(reader.Attribute("stockAmount").Value)
            };
// 本の項目ソースを設定します
book.ItemsSource = books;
}
#region Object Model
public Orientation Orientation
{
    get
    {
        return book.Orientation;
    }
    set
    {
        book.Orientation = value;
    }
}
public bool IsFirstPageOnTheRight
{
    get
    {
        return book.IsFirstPageOnTheRight;
    }
    set
    {
        book.IsFirstPageOnTheRight = value;
    }
}
public PageFoldVisibility ShowPageFold
{
    get
    {
        return book.ShowPageFold;
    }
    set
    {
        book.ShowPageFold = value;
    }
}
public PageFoldAction PageFoldAction
{
    get
```



```

        {
            return book.PageFoldAction;
        }
        set
        {
            book.PageFoldAction = value;
        }
    }
    public double FoldSize
    {
        get
        {
            return book.FoldSize;
        }
        set
        {
            book.FoldSize = value;
        }
    }
    public int CurrentPage
    {
        get
        {
            return book.CurrentPage;
        }
        set
        {
            book.CurrentPage = value;
        }
    }
}
#endregion

```

この手順では、C1Book コントロールにテンプレートを追加し、別のファイルからコンテンツを呼び出すコードを追加しました。次の手順では、アプリケーションにいくつかのファイルを追加します。

手順3: アプリケーションへのファイルの追加

C1Book コントロールにテンプレートとコードを追加しました。この手順ではアプリケーションにいくつかのファイルを追加します。これらのコンテンツファイルは、データを表示するためにコードから呼び出されます。

1. プロジェクト名を右クリックし、メニューから**[追加]**→**[新しい項目]**を選択します。
2. **[新しい項目の追加]** ウィンドウで、**[XML ファイル]**を選択します。ファイルに Amazon.xml と名前を付けます。ファイルが自動的に表示されます。
3. 次のマークアップを Amazon.xml ファイルに追加します。

▶ XAML でマークアップの書き方

マークアップ

```

<books>
  <book id="0672328917" coverUri="http://www.coverbrowser.com/image/bestsellers-2007/1943-1.jpg" price="$49.99" title="Windows Presentation Foundation Unleashed (WPF) (Unleashed)" stockAmount="1" />

```

```
<!--
<book id="073562528X"
coverUri="http://borntolearn1.mslearn.net/images/2009/06/9780735625730f.jpg"
price="$34.99" title="Introducing Microsoft Silverlight 3.0" stockAmount="200"
/>
  <book id="0596510373" coverUri="http://ecx.images-
amazon.com/images/I/51DF0boY5fL.jpg" price="$49.99" title="Programming WPF"
stockAmount="30" />
  -->
  <book id="0596527438" coverUri="http://www.coverbrowser.com/image/oreilly-
books/97-1.jpg" price="$49.99" title="Programming C# 3.0 (Programming)"
stockAmount="5" />
  <book id="0596519982" coverUri="http://ecx.images-
amazon.com/images/I/51U9eZeXhuL_SL500_.jpg" price="$34.99" title="Essential
Silverlight 2 Up-to-Date (Up-To-Date)" stockAmount="8" />
  <book id="1590599594" coverUri="http://ecx.images-
amazon.com/images/I/51DedRUoZGL.jpg" price="$44.99" title="Beginning Web
Development, Silverlight, and ASP.NET AJAX: From Novice to Professional
(Beginning from Novice to Professional)" stockAmount="178" />
  <book id="059651509X" coverUri="http://www.coverbrowser.com/image/oreilly-
books/30-1.jpg" price="$44.99" title="Painting the Web" stockAmount="1" />
  <book id="0672330075" coverUri="http://vig-
fp.pearsoned.co.uk/bigcovers/0672330075.jpg" price="$39.99" title="Silverlight
1.0 Unleashed" stockAmount="1" />
  <book id="0672329689" coverUri="http://images.pearsoned-
ema.com/jpeg/large/9780672329685.jpg" price="$34.99" title="Creating Vista
Gadgets: Using HTML, CSS and JavaScript with Examples in RSS, Ajax, ActiveX
(COM) and Silverlight" stockAmount="1" />
  <book id="1590599764" coverUri="http://knowfree.net/wp-
content/uploads/2008/05/1590599764011-250x299.jpg" price="$39.99"
title="Foundation Expression Blend 2: Building Applications in WPF and
Silverlight (Foundation)" stockAmount="13" />
  <!--
  <book id="1590599497" coverUri="http://i39.tinypic.com/dqhnoi.jpg"
price="$44.99" title="Pro Silverlight 2 in C# 2008 (Windows.Net)"
stockAmount="1" />
  <book id="159059939X"
coverUri="http://www.sql163.com/UploadFile/Book_Image/2009-
3/Sql163_2009329192012.jpg" price="$14.99" title="Silverlight and ASP.NET
Revealed" stockAmount="1" />
  -->
</books>
```

- プロジェクト名を再度右クリックし、リストから[追加]→[新しい項目]を選択します。[新しい項目の追加]ウィンドウが開いたら、[コードファイル]を選択し、AmazonBookDescription.cs と名前を付けます。
- 新しいコードファイルが自動的に表示されます。次のコードをファイルに追加します。

▶ C# コードの書き方

```
C#
using System;
using System.Collections.Generic;
using System.Linq;
```

```

using System.Text;
using System.Windows;
using System.Windows.Input;
namespace BookTest
{
    public class AmazonBookDescription
    {
        public string Title { get; set; }
        public string CoverUri { get; set; }
        public string Id { get; set; }
        public string Price { get; set; }
        public int StockAmount { get; set; }
    }
}

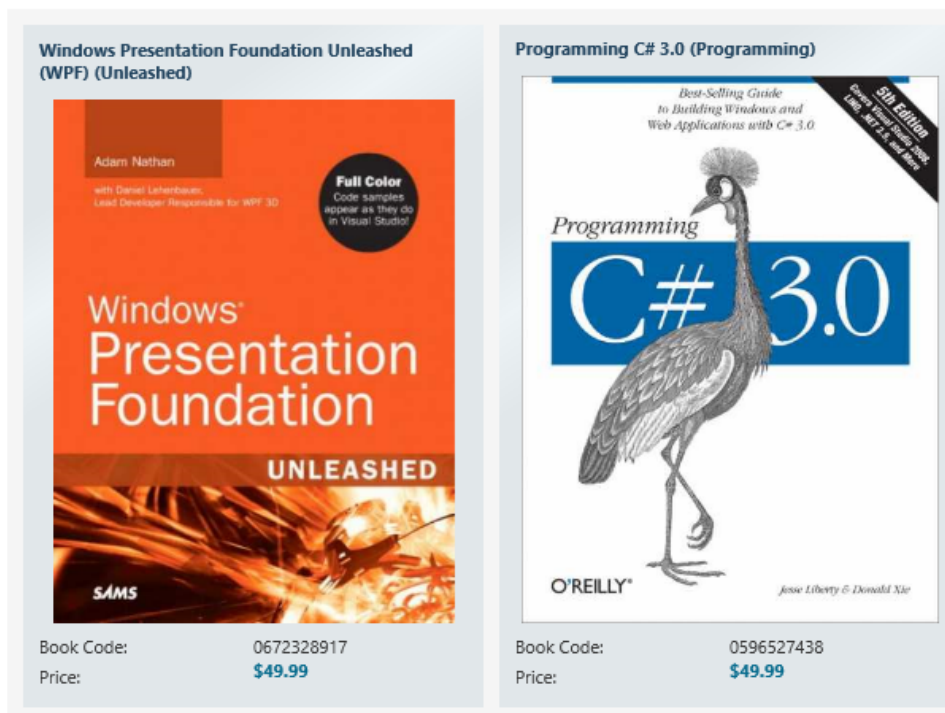
```

この手順では、2つのファイルをアプリケーションに追加しました。次の手順では、このアプリケーションを実行します。

手順4: Book アプリケーションの実行

アプリケーションを作成できたので、後はアプリケーションを実行するだけです。アプリケーションを実行し、**Book for UWP** の実行時の動作を確認するには、次の手順に従います。

1. **[プロジェクト]**メニューから**[プロジェクトの実行]**を選択して、実行時にアプリケーションがどのように表示されるかを確認します。
アプリケーションは次の図のように表示されます。



最初に1ページしか表示されないように **IsFirstPageOnTheRight** プロパティを設定しました。C1Book コントロールの右下または右上にカーソルを置くと、ページが少し折れて、ページをめくることができることがわかります。詳細については、「**ブックのゾーン**」を参照してください。

2. ページの右上をタップするとページがめくれ、2ページ目と3ページ目が表示されます。

<p>Essential Silverlight 2 Up-to-Date (Up-To-Date)</p> <p>Book Code: 0596519982 Price: \$34.99</p>	<p>Beginning Web Development, Silverlight, and ASP.NET AJAX: From Novice to Professional (Beginning from Novice to Professional)</p>  <p>Book Code: 1590599594 Price: \$44.99</p>
--	---

おめでとうございます。

これで **Book for UWP** クイックスタートは完了です。簡単なアプリケーションを作成し、**Book for UWP** コントロールを1つ追加してカスタマイズしました。その後、コントロールの実行時機能をいくつか確認しました。

C1Book の使い方

Book for UWP には **C1Book** コントロールが入っています。これは、コンテナとして動作する簡単なブックコントロールです。コントロールや画像などを見慣れた本の形式で追加できます。XAML ウィンドウに追加された C1Book コントロールは、パネルに似たコンテナになります。これをカスタマイズしたり、これにコンテンツを追加することができます。

コントロールのインターフェースは、次の図のように表示されます。



基本的なプロパティ

Book for UWP には、コントロールの機能を設定するためのいくつかのプロパティがあります。主要なプロパティを次に示します。外観を制御するプロパティの詳細については、「**外観プロパティ**」を参照してください。

次のプロパティを使用して、C1Book コントロールをカスタマイズできます。

プロパティ	説明
CurrentPage	現在表示されているページを取得または設定します。
CurrentZone	マウスが置かれているブックのゾーンを取得します。詳細については、「 ブックのゾーン 」を参照してください。
FoldSize	折り返しのサイズをピクセル単位で取得または設定します。詳細については、「 ページの折り返しのサイズ 」を参照してください。
IsFirstPageOnTheRight	ブックの最初のページがブックの右側に表示されるかどうかを取得または設定します。詳細については、「 最初のページの表示 」を参照してください。
IsMouseOver	マウスがコントロール内にある場合は、 True を返します。
LeftPageTemplate	ブックの左側に表示されるページのテンプレートを取得または設定します。
PageFoldAction	ページめくりアニメーションを発生させるアクションを取得または設定します。詳細については、「 ページめくりオプション 」を参照してください。
RightPageTemplate	ブックの右側に表示されるページのテンプレートを取得または設定します。
ShowPageFold	折り返しを表示するかどうかを取得または設定します。詳細については、「 ページの折り返しの表示/非表示 」を参照してください。
TurnInterval	ページめくりアニメーションの時間の長さ(ミリ秒)を取得または設定します。

基本的なイベント

Book for UWP には、コントロールの操作を設定およびカスタマイズするためのイベントがあります。主要なイベントを次に示します。

次のイベントを使用して、**C1Book** コントロールをカスタマイズできます。

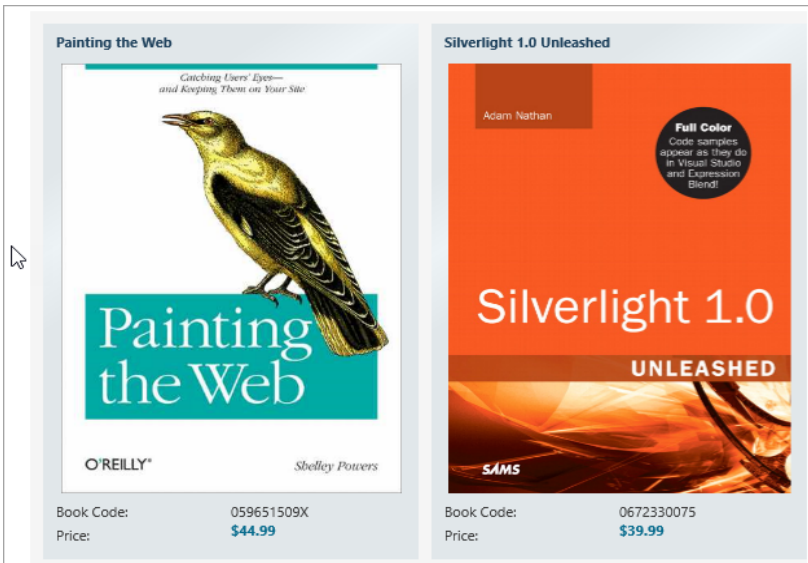
イベント	説明
CurrentPageChanged	CurrentPage プロパティが変更されると発生します。
CurrentZoneChanged	現在のゾーンが変更されると発生します。詳細については、「 ブックのゾーン 」を参照してください。
DragPageFinished	ページのドラッグの終了時に発生します。
DragPageStarted	ページのドラッグの開始時に発生します。
IsMouseOverChanged	IsMouseOver プロパティが変更されたときに発生するイベントです。

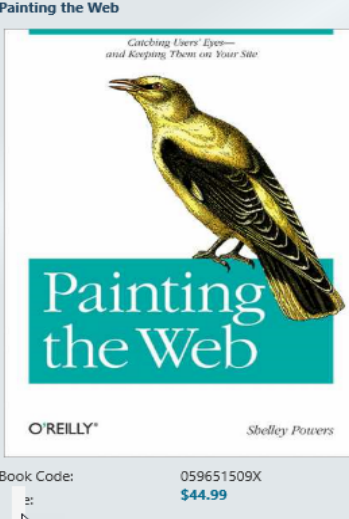
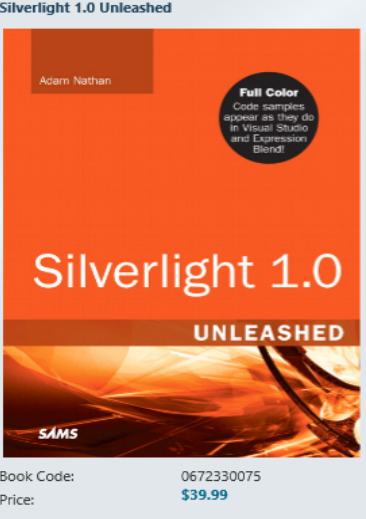
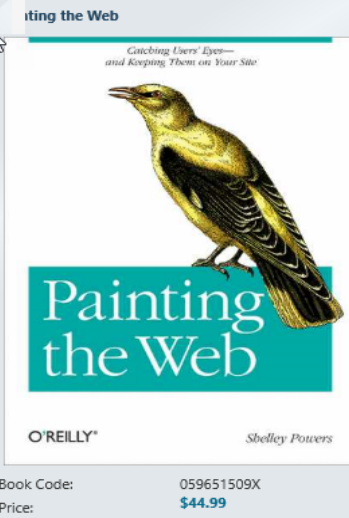
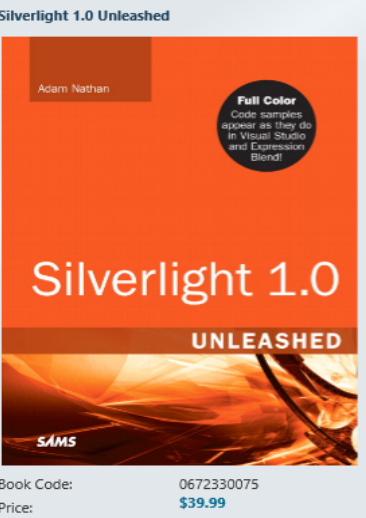
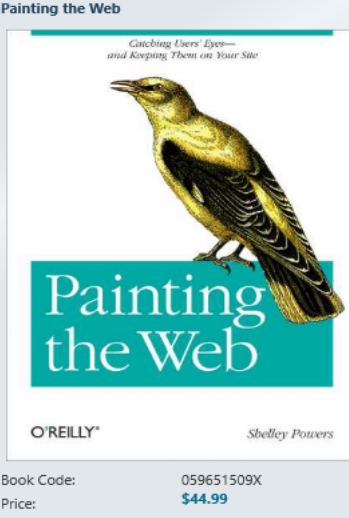
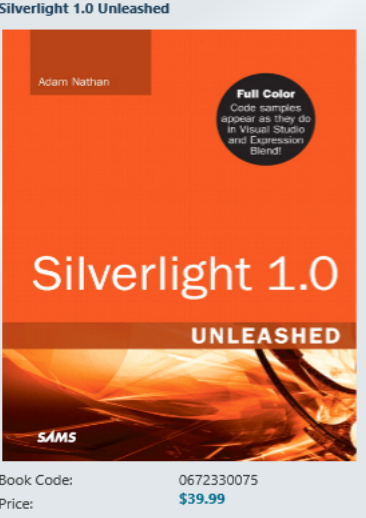
ブックのゾーン

C1Book コントロールにはいくつかのゾーンがあります。これらのゾーンを使用して、ユーザーがコントロールの各部を操作したときに行う動作をカスタマイズできます。**CurrentZone** プロパティを使用すると、現在のゾーンを取得できます。

CurrentZoneChanged イベントを使用すると、ユーザーが別のゾーンに移動したときに行う動作をカスタマイズできます。

C1Book コントロールには6つのゾーンがあります。各ゾーンの具体的な場所については、次の表内の画像でマウスの位置を確認してください。

ゾーン	説明	例
Out	ブックの境界線の外側のゾーンを指定します。	

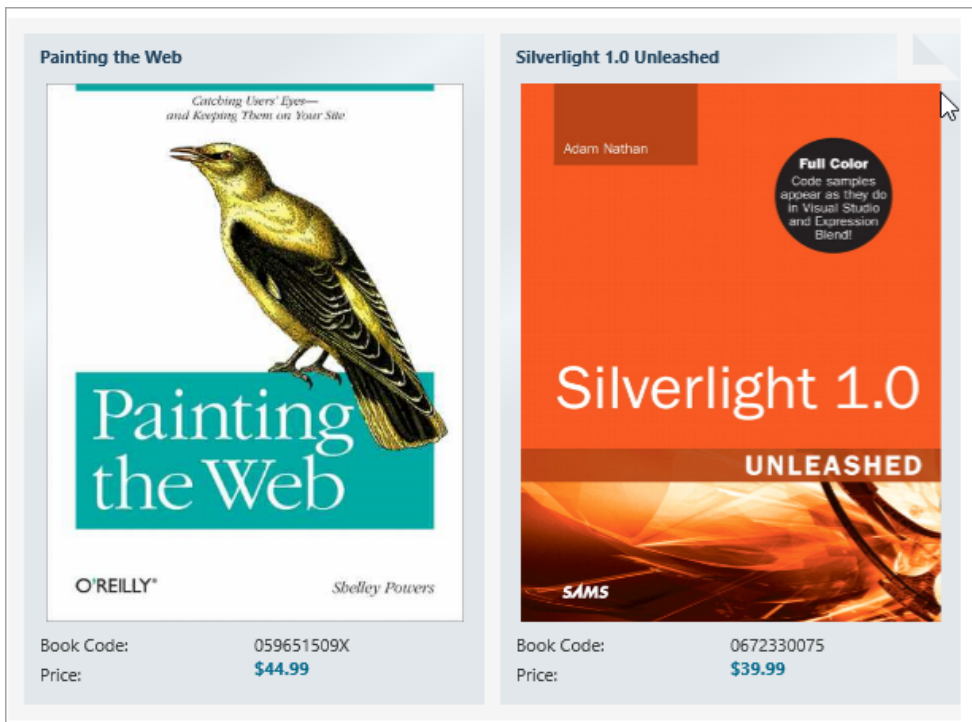
<p>BottomLeft</p>	<p>左下の折り返しゾーンを指定します。</p>		
<p>TopLeft</p>	<p>左上の折り返しゾーンを指定します。</p>		
<p>Center</p>	<p>ブックの中央を指定します (折り返しゾーンではない)。</p>		

TopRight	右上の折り返しゾーンを指定します。	
BottomRight	右下の折り返しゾーンを指定します。	

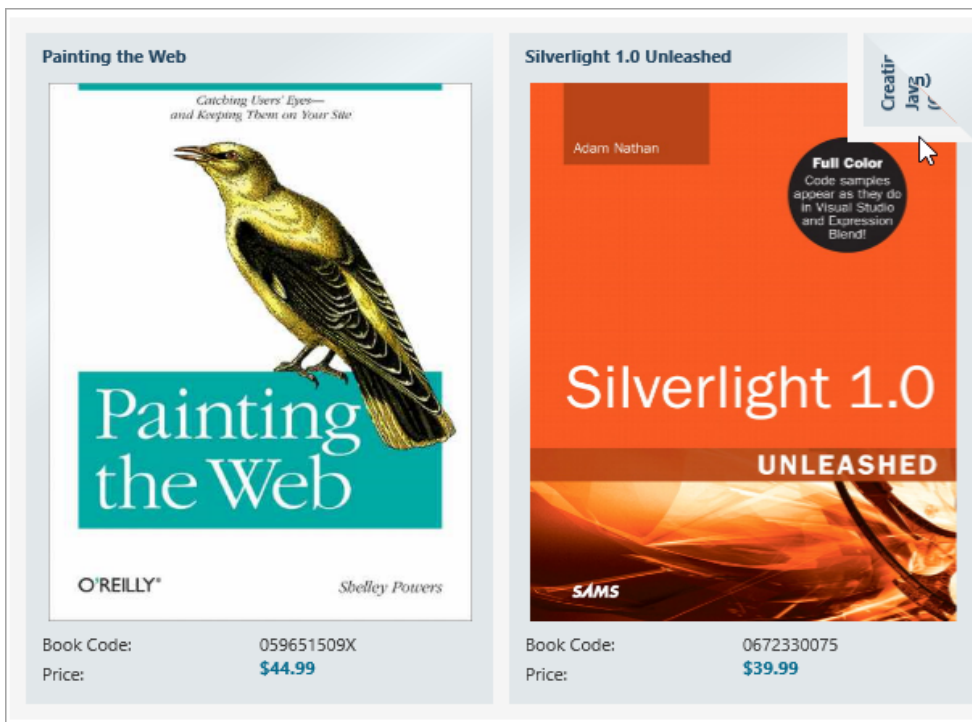
ページの折り返しのサイズ

ブックのページをめくる際の外観をカスタマイズする方法の1つは、**FoldSize** プロパティを使用して、ページの折り返しのサイズを設定することです。ページの折り返しは、ユーザーが所定の**ブックのゾーン**にマウスを置くと表示され、ページをめくることができることを示します。

FoldSize プロパティを設定すると、すべてのページ折り返し(右上、右下、左上、左下)のサイズが設定されます。例えば、FoldSize を **40** に設定すると、すべての折り返しが次の図のように表示されます。



大きい数字を設定するほど、折り返しも大きくなります。例えば、FoldSize を **80** に設定すると、すべての折り返しが次の図のように表示されます。



ページの折り返しの表示/非表示

C1Book を使用すると、ページの折り返しの表示/非表示を設定することができます。デフォルトでは、ユーザーが所定のブックゾーンにマウスを置くと、ページの折り返しが表示されます。ただし、必要に応じて、ページの折り返しの表示/非表示を変更できます。**ShowPageFold** コントロールを次のように設定できます。

値	説明
OnMouseOver	折り返しは、ユーザーがマウスでページの端をドラッグすると表示されます。これはデフォルト設定です。

Never	折り返しは表示されません。
Always	折り返しは常に表示されます。

ページめくりオプション

ユーザーがページの折り返しを1回タップすると、ブックに前のページまたは次のページが表示されます。これはC1Bookのデフォルトの動作です。**PageFoldAction** プロパティを使用して、ページをめくる方法をカスタマイズできます。たとえば、ユーザーがダブルタップするとページがめくられるように PageFoldAction を設定できます。または、マウスのクリックではページをめくらず、ページの折り返し上でスライド操作をしないとページめくりが行われなくともできます。

PageFoldAction コントロールを次のように設定できます。

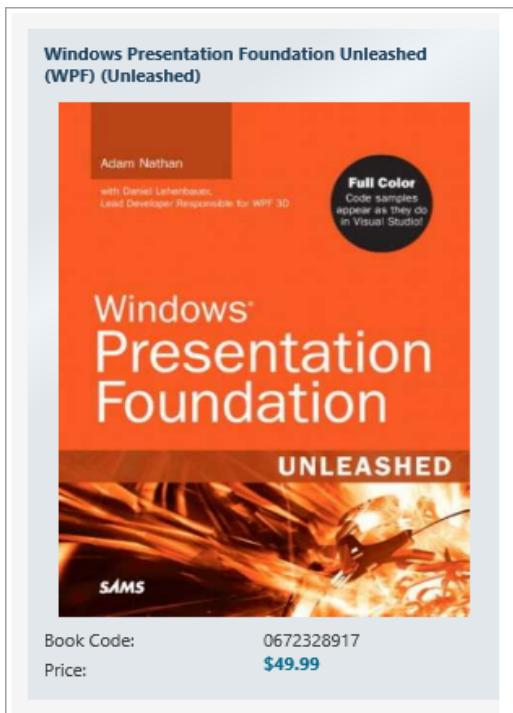
値	説明
TurnPageOnClick	ユーザーがページの折り返しをクリックするとページがめくられます。
TurnPageOnDoubleClick	ユーザーがページの折り返しをダブルクリックするとページがめくられます。
None	ユーザーがページの折り返しをブック内で左右にドラッグするとページがめくられます。

最初のページの表示

デフォルトでは、**C1Book** コントロールの最初のページは左側に表示されます。これは、開いた本のような見え方になります。




また、**IsFirstPageOnTheRight** プロパティを **True** に設定すると、最初のページが右側に表示されるように変更できます。最初のページが右側に表示されるように設定すると、閉じた本の表紙のように見えます。



ブックナビゲーション

実行時は、ジェスチャを使用して **C1Book** コントロール内を移動できます。ブックゾーン内でタップしたり、スライド操作を行ってページをめくります。C1Book コントロールには、ナビゲーション関連のメソッド、プロパティ、およびイベントが含まれており、現在どのページが表示されているかを簡単に判断したり、ユーザーがブック内を移動する際のアプリケーションのアクションを簡単に設定することができます。

- **CurrentPage** プロパティは、実行時に現在表示中のページを取得または設定します。

 **注意:** ページをめくると、見開きで左側に表示されるページが **CurrentPage** になることに注意してください。ページ番号は **0** から始まり、ページ **0** は常に左側に表示されます。したがって、**IsFirstPageOnTheRight** プロパティを **True** に設定すると、ブックの最初のページは最初に右側に表示されてページ1になり、左側には非表示のページ **0** があります。

- 表示されるページは **CurrentPage** プロパティを使用して設定できますが、**TurnPage** メソッドを使用して、実行時に現在のページを変更することもできます。**TurnPage** メソッドは、ブックを1ページずつ進めたり戻したりします。
- **CurrentPageChanged** イベントを使用すると、現在のページが変更されたときに発生するアクションを指定できます。
- **DragPageStarted** イベントと **DragPageFinished** イベントを使用すると、ユーザーがスライド操作によってページをめくったときに発生するアクションを指定できます。

レイアウトおよび外観

以下のトピックでは、**C1Book** コントロールのレイアウトと外観をカスタマイズする方法について詳しく説明します。

色のプロパティ

次のプロパティを使用して、コントロール自体に使用される色をカスタマイズできます。

プロパティ	説明
Background	コントロールの背景を描画するブラシを取得または設定します。これは依存プロパティです。

Foreground | 前景色を描画するブラシを取得または設定します。これは依存プロパティです。

境界線のプロパティ

次のプロパティを使用して、コントロールの境界線をカスタマイズできます。

プロパティ	説明
BorderBrush	コントロールの境界線の背景を描画するブラシを取得または設定します。これは依存プロパティです。
BorderThickness	コントロールの境界線の太さを取得または設定します。これは依存プロパティです。

サイズのプロパティ

次のプロパティを使用すると、コントロールのサイズをカスタマイズできます。

プロパティ	説明
Height	要素の推奨高さを取得または設定します。これは依存プロパティです。
MaxHeight	要素の最大高さ制約を取得または設定します。これは依存プロパティです。
MaxWidth	要素の最大幅制約を取得または設定します。これは依存プロパティです。
MinHeight	要素の最小高さ制約を取得または設定します。これは依存プロパティです。
MinWidth	要素の最小幅制約を取得または設定します。これは依存プロパティです。
Width	要素の幅を取得または設定します。これは依存プロパティです。

ブックのスタイル

Book for UWP の C1Book コントロールは、コントロールの外観を変更するために使用できるスタイルのプロパティを提供します。これらのスタイルの一部について、次の表で説明します。

スタイル	説明
FontStyle	フォントスタイルを取得または設定します。これは依存プロパティです。
Style	この要素のレンダリング時に使用されるスタイルを取得または設定します。これは依存プロパティです。

タスク別ヘルプ

次のタスク別ヘルプトピックは、ユーザーの皆様が Visual Studio に精通しており、**C1Book** コントロールの一般的な使用方法を理解していることを前提としています。**Book for UWP** 製品を初めて使用される場合は、まず「[クイックスタート](#)」を参照してください。

このセクションの各トピックは、**Book for UWP** 製品を使用して特定のタスクを実行するための方法を提供します。タスク別ヘルプの多くのトピックは、新しい空の Windows ストアアプリケーションが作成されており、プロジェクトに適切な参照と C1Book コントロールが追加されていることを前提としています。コントロールの作成の詳細については、「[ブックの作成](#)」を参照してください。

ブックの作成

C1Book コントロールは、設計時、XAML、およびコードで簡単に作成できます。次の手順で作成した C1Book コントロールは、空のコンテナとして表示されます。このコントロールを実行時にブックとして表示するには、項目を追加する必要があります。例については、「[ブックへの項目の追加](#)」を参照してください。

XAML の場合

C1Book コントロールを XAML マークアップを使用して作成するには、次の手順に従います。

1. Visual Studio Solution Explorer で、プロジェクトファイルリスト内の [**参照**] フォルダを右クリックします。コンテキストメニューから [**参照の追加**] を選択し、**ComponentOne for UWP** アセンブリを選択して、[OK] をクリックします。
2. `<Grid>` タグと `</Grid>` タグの間にカーソルを置きます。Visual Studio のツールボックスで C1Book コントロールをダブルクリックしてアプリケーションに追加します。これで、`<Page>` タグに次の名前空間も追加されます。

マークアップ

```
xmlns:Extended="using:C1.Xaml.Extended"
```

3. 次のマークアップを `<Extended: C1Book>` タグに追加します。
 - `Name="c1book1"`
 - `Height="300"`
 - `Width="450"`

このマークアップは、「c1book1」という名前の空の C1Book コントロールを作成し、コントロールのサイズを設定します。

コードの場合

C1Book コントロールをコードで作成するには、次の手順に従います。

1. Visual Studio Solution Explorer で、プロジェクトファイルリスト内の [**参照**] フォルダを右クリックします。コンテキストメニューから [**参照の追加**] を選択し、**ComponentOne for UWP** アセンブリを選択して、[OK] をクリックします。
2. **MainPage.xaml** ウィンドウで、`<Grid>` タグ内にカーソルを置き、次のマークアップを追加します。
 - `Name="Grid1"`
3. [**MainPage.xaml**] ウィンドウ内で右クリックし、[**コードの表示**] を選択してコードビューに切り替えます。
4. 次の `import` 文をページの先頭に追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
Imports C1.Xaml.Extended
```

▶ C# コードの書き方

C#

```
using C1.Xaml.Extended;
```

5. ページのコンストラクタに、C1Book コントロールを作成するコードを追加します。次のようになります。

▶ Visual Basic コードの書き方

Visual Basic

```
Public Sub New()
    InitializeComponent()
    Dim clbook1 as New C1Book
    clbook1.Height = 300
    clbook1.Width = 450
    Grid1.Children.Add(clbook1)
End Sub
```

▶ C# コードの書き方

```
C#
public MainPage ()
{
    this.InitializeComponent ();
    C1Book clbook1 = new C1Book ();
    clbook1.Height = 300;
    clbook1.Width = 450;
    Grid1.Children.Add (clbook1);
}
```

このコードは、「clbook1」という名前の空の C1Book コントロールを作成し、コントロールのサイズを設定し、コントロールをページに追加します。

🟢 ここまでの成果

C1Book コントロールを作成しました。上記の手順で作成した C1Book コントロールは、空のコンテナとして表示されます。このコントロールを実行時にブックとして表示するには、項目を追加する必要があります。例については、「[ブックへの項目の追加](#)」を参照してください。

ブックへの項目の追加

C1Book コントロールには、任意の種類コンテンツを追加できます。これには、テキスト、画像、レイアウトパネルなどの標準コントロールやサードパーティのコントロールがあります。この例では、C1Book コントロールに **TextBlock** コントロールを追加しますが、手順をカスタマイズして他の種類のコンテンツを追加することもできます。

XAML の場合

たとえば、**TextBlock** コントロールをブックに追加するには、`<TextBlock Text="Hello World!"/>` を `<Extended:C1Book>` タグ内に追加します。次のようになります。

マークアップ

```
<Extended:C1Book Name="clbook1" Height="300" Width="450">
    <TextBlock Text="Hello World!"/>
</Extended:C1Book>
```

コードの場合

たとえば、**TextBlock** コントロールをブックに追加するには、ページのコンストラクタに次のようにコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
Public Sub New ()
    InitializeComponent ()
Dim clbook1 as New C1Book
    clbook1.Height = 300
    clbook1.Width = 450
    Grid1.Children.Add (clbook1)
Dim txt1 as New TextBlock
    txt1.Text = "Hello World!"
    clbook1.Items.Add (txt1)
```


End Sub

▶ C# コードの書き方

```
C#
public MainPage ()
{
    this.InitializeComponent ();
    C1Book clbook1 = new C1Book ();
    clbook1.Height = 300;
    clbook1.Width = 450;
    Grid1.Children.Add (clbook1);
    TextBlock txt1 = new TextBlock ();
    txt1.Text = "Hello World!";
    clbook1.Items.Add (txt1);
}
```

✔ ここまでの成果

C1Book コントロールにコントロールを追加しました。アプリケーションを実行し、**TextBlock** コントロールが C1Book コントロールに追加されていることを確認してください。他のコンテンツやコントロールも同様に追加できます。

ブック内の項目のクリア

たとえば、実行時にユーザーがすべての項目を **C1Book** コントロールからクリアできるようにすることができます。また、コントロールを連結してから別のデータソースに再連結する際に、項目コレクションをクリアすることもできます。

たとえば、ブックのコンテンツをクリアするには、次のコードをプロジェクトに追加します。

▶ Visual Basic コードの書き方

```
Visual Basic
clbook1.Items.Clear ();
```

▶ C# コードの書き方

```
C#
clbook1.Items.Clear ();
```

✔ ここまでの成果

コントロールのコンテンツがクリアされます。アプリケーションを実行すると、ブックが空になっています。

最初のページを右側に表示

IsFirstPageOnTheRight プロパティは、最初のページを右側と左側のどちらに表示するかを定義します。詳細については、「[最初のページの表示](#)」を参照してください。デフォルトでは、C1Book コントロールは、起動時に最初のページを左側にして2ページを見開きで表示します。これは、XAML およびコードで **IsFirstPageOnTheRight** プロパティを設定してカスタマイズできます。

XAML の場合

たとえば、**IsFirstPageOnTheRight** プロパティを設定するには、**IsFirstPageOnTheRight="True"** を

Extended Library for UWP

<Extended:C1Book> タグに追加します。次のようになります。

マークアップ

```
<Extended:C1Book Name="c1book1" Height="300" Width="450"
IsFirstPageOnTheRight="True">
```

コードの場合

たとえば、IsFirstPageOnTheRight プロパティを設定するには、ページのコンストラクタで次のコードをプロジェクトに追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
c1book1.IsFirstPageOnTheRight = True
```

▶ C# コードの書き方

C#

```
c1book1.IsFirstPageOnTheRight = true;
```

🟢 ここまでの成果

最初のページが右側に表示されるように設定しました。アプリケーションを実行すると、最初のページが本の表紙のように1ページだけ表示されます。



現在のページの設定

CurrentPage プロパティは、**C1Book** コントロールの現在のページの値を取得または設定します。デフォルトでは、C1Book コントロールは、起動時に最初のページを表示します。これは、XAML およびコードで CurrentPage プロパティを設定してカスタマイズできます。

XAML の場合

たとえば、CurrentPage プロパティを **3** に設定するには、**CurrentPage="3"** を <Extended:C1Book> タグに追加します。次のようになります。

マークアップ

```
<Extended:C1Book x:Name="clbook1" Height="300" Width="450" CurrentPage="3">
```

コードの場合

たとえば、CurrentPage プロパティを **3** に設定するには、プロジェクトに次のコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
clbook1.CurrentPage = 3
```

▶ C# コードの書き方

C#

```
clbook1.CurrentPage = 3;
```

 ここまでの成果

ブックの初期開始ページを変更しました。アプリケーションを実行すると、最初にページ3が表示されます。

コードでのブック内のナビゲーション

表示されるページは **CurrentPage** プロパティを使用して設定できますが、**TurnPage** メソッドを使用して、実行時に現在のページを変更することもできます。詳細については、「[ブックナビゲーション](#)」を参照してください。このトピックでは、アプリケーションに2つのボタンを追加します。一方はページをめくって前のページに戻り、もう一方はページをめくって次のページに進みます。

Blend でブックにナビゲーションを追加するには、次の手順に従います。

1. C1Book コントロールの名前を「**c1book1**」に設定し、`<Extended:C1Book>` タグと `</Extended:C1Book>` タグの間に次の XAML マークアップを追加します。これで、チェッカー盤のような6つのページがアプリケーションに追加されます。

▶ XAML でマークアップの書き方

マークアップ

```
<Grid Name="checkers" Background="White" >
  <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 Name="checkers2" Background="White">
  <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="checkers3" Background="White" >
  <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 Name="checkers4" Background="White" >
    <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="checkers5" Background="White" >

```

```

<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 Name="checkers6" Background="White" >
    <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" />

```

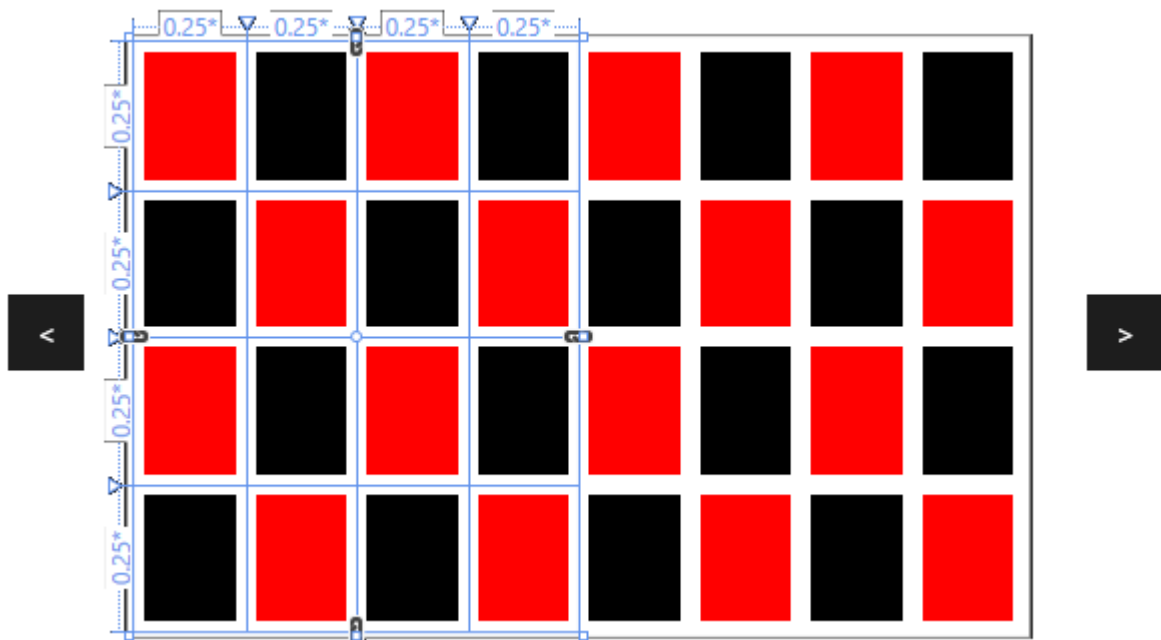
```
<Rectangle Fill="Black" Grid.Row="2" Grid.Column="3" Margin="5" />
<Rectangle Fill="Black" Grid.Row="3" Grid.Column="0" Margin="5" />
<Rectangle Fill="Red" Grid.Row="3" Grid.Column="1" Margin="5" />
<Rectangle Fill="Black" Grid.Row="3" Grid.Column="2" Margin="5" />
<Rectangle Fill="Red" Grid.Row="3"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

2. ツールボックスに移動し、**[ボタン]**項目をダブルクリックして、アプリケーションに2つの **Button** コントロールを追加します。
3. 1つめのボタンを選択し、**[プロパティ]**ウィンドウに移動して、次のプロパティを設定します。
 - **[Name]**を「btn_last」に
 - **[Content]**を「<」に
 - **[Height]**と**[Width]**を「48」に
4. 2つめのボタンを選択し、**[プロパティ]**ウィンドウに移動して、次のプロパティを設定します。
 - **[Name]**を「btn_next」に
 - **[Content]**を「>」に
 - **[Height]**と**[Width]**を「48」に
5. **btn_last** ボタンをブックの左側に、**btn_next** ボタンをブックの右側に置いて、ボタンの位置を変更します。
6. 左ボタンをダブルクリックして、**Click** イベントを作成します。
7. デザインビューに戻り、同じ手順を右のボタンで繰り返します。つまり、各ボタンに Click イベントを指定します。XAML マークアップは次のようになります。

マークアップ

```
<Button x:Name="btn_last" HorizontalAlignment="Left" Margin="49,223,0,229"
Width="48" Height="48" Content="&lt;" Click="btn_last_Click"/>
<Button x:Name="btn_next" HorizontalAlignment="Right" Margin="0,224,49,228"
Width="48" Height="48" Content="&gt;" Click="btn_next_Click"/>
```

ページは次のようになります。



8. コードビューに切り替え、次の **import** 文をページの先頭に追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
Imports Cl.Xaml.Extended
```

▶ C# コードの書き方

C#

```
using Cl.Xaml;  
using Cl.Xaml.Extended;
```

9. **Click** イベントハンドラにコードを追加します。次のようになります。

▶ Visual Basic コードの書き方

Visual Basic

```
Private Sub btn_next_Click(ByVal sender as Object, ByVal e as  
System.Windows.RoutedEventArgs)  
    clbook1.TurnPage(True)  
End Sub  
Private Sub btn_last_Click(ByVal sender as Object, ByVal e as  
System.Windows.RoutedEventArgs)  
    clbook1.TurnPage(False)  
End Sub
```

▶ C# コードの書き方

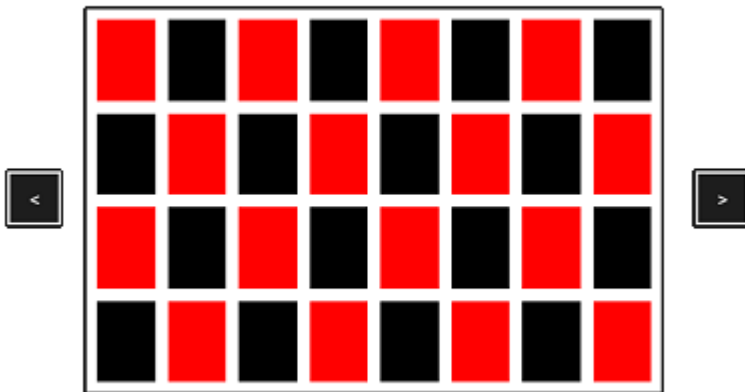
C#

```
public MainPage()  
{  
    private void btn_next_Click(object sender, System.Windows.RoutedEventArgs e)  
    {  
        clbook1.TurnPage(true);  
    }  
    private void btn_last_Click(object sender, System.Windows.RoutedEventArgs e)  
    {  
        clbook1.TurnPage(false);  
    }  
}}
```

このコードは、ボタンのタップによってブックを1ページ前または後にめくります。

✔ ここまでの成果

ブック内のナビゲーションをカスタマイズしました。ブック内のナビゲーションを確認するには、アプリケーションを実行し、右ボタンをタップします。ページめくりアニメーションによって次のページがめくられることがわかります。



左ボタンをタップし、ブックが前のページに戻ることを確認します

ColorPicker for UWP

ColorPicker for UWP は、ユーザーに機能豊富な対話式の色選択インターフェースを提供する色入力エディタです。本格的なデザインのパレットから色を選択することも、カスタム色を選択することもできます。ColorPicker for UWP では、標準の色オプションが用意された **Basic** カラーパレット、ユーザーが色の選択をカスタマイズできる **Advanced** パレット、またはその両方を使用するように選択できます。

C1ColorPicker クラスは、強力でビジュアルな色入力インターフェースを提供するために、C1SpectrumColorPicker、16 進数色、RGB 色もサポートしています。

主な特長

ColorPicker for UWP を使用すると、マルチカラーパレットでカスタマイズされた、見た目にも優れたアプリケーションを作成できます。**ColorPicker for UWP** は、サポートされているデバイスで使いやすいように、はっきりとした大きなボタンとアイコンで構成されています。

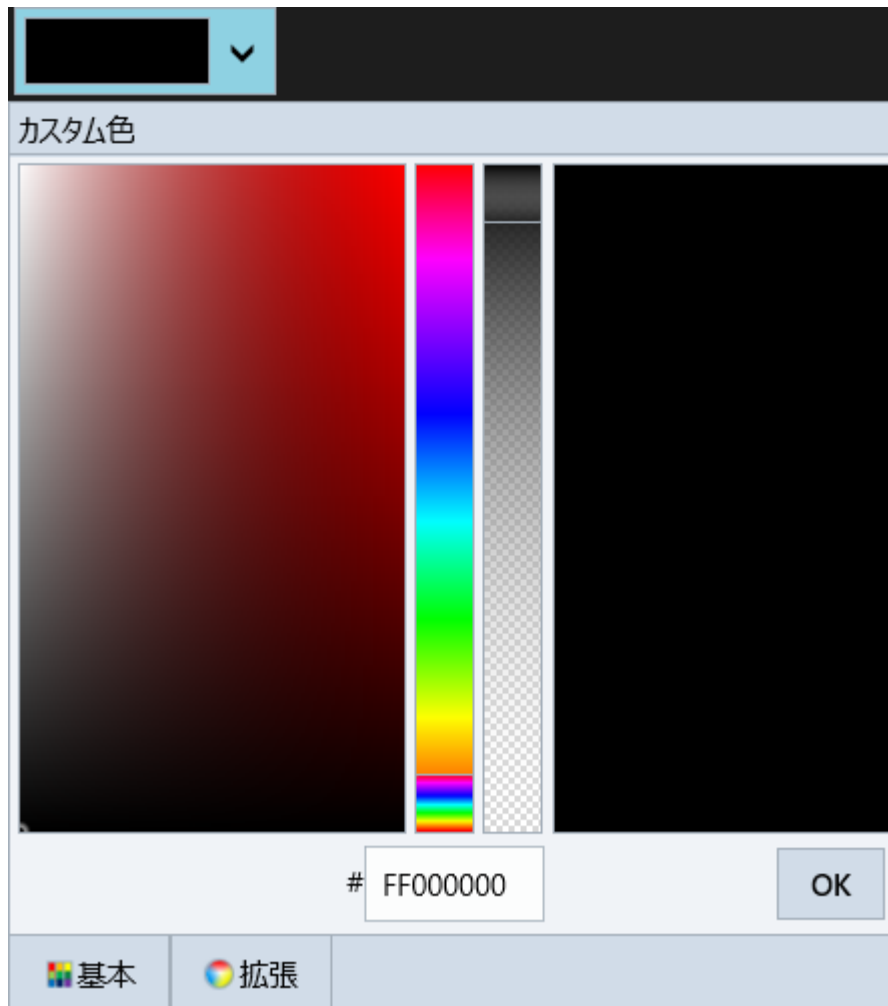
- **20 以上の本格的なデザインのカラーパレットを選択可能**

C1ColorPicker には、Microsoft Office スイートで使用されるテーマに合わせた 20 以上のカラーパレットが定義されています。各パレットにある色は互いによく調和しているため、洗練されたプロフェッショナルなアプリケーションを作成に役立ちます。



- **カスタム色を選択できる組み込みカラーエディタ**

C1ColorPicker には、カラーエディタも用意されています。カラーエディタを使用して、標準のカラーパレットにはないカスタム色を作成できます。このエディタには、RGB モデルを使用したり、選択した色を透過と不透過でカスタマイズする機能もあります。これらの機能は、Visual Studio プロジェクトで、C1ColorPicker の[プロパティ]ウィンドウの **Background** プロパティに表示されています。



- 複数のモード

C1ColorPicker は、実行時の色選択で **Basic** モードと **Advanced** モードをサポートします。

- 構成可能なパーツ

C1ColorPicker には、色の選択時のカスタマイズをサポートする 3 つの追加コントロール (**C1SpectrumColorPicker**、**C1HexColorBox**、**C1CheckedBorder**) が含まれます。**C1SpectrumColorPicker** コントロールは、高度な色選択機能を利用できるようにします。**C1HexColorBox** コントロールは、16 進コードで入力されるデータを検証するために使用されます。**C1CheckedBorder** は、透過性の異なる色を表示するために追加されています。

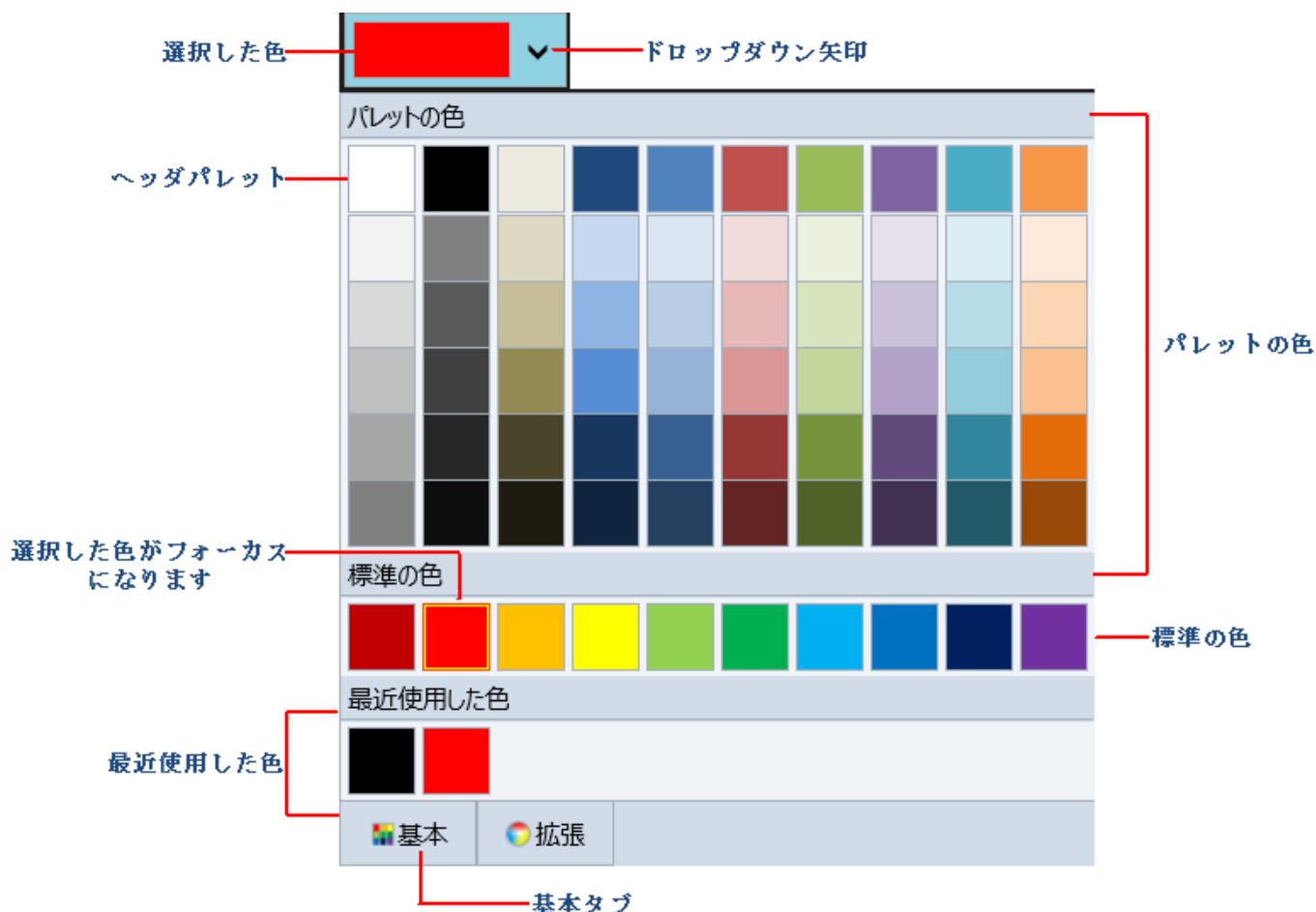
ビジュアル要素

ColorPicker for UWP を使用すると、マルチカラーパレットでカスタマイズされた、見た目にも優れたアプリケーションを作成できます。次の図は、**C1ColorPicker** のさまざまな要素を示します。

基本モード

C1ColorPicker の **Mode** プロパティを使用すると、事前に選択されたカラーパレットから色を選択することも、必要に応じて独自のパレットをカスタマイズすることもできます。C1ColorPicker は 3 つのモード (**Basic**、**Advanced**、**Both**) をサポートします。デフォルトでは、Mode プロパティは **Both** に設定され、Basic と Advanced の両方のカラーパレットが表示されます。

基本モード: デフォルトでは、C1ColorPicker のドロップダウン矢印がクリックされると、[基本] タブが開かれます。[基本] タブは、次の図のように表示されます。

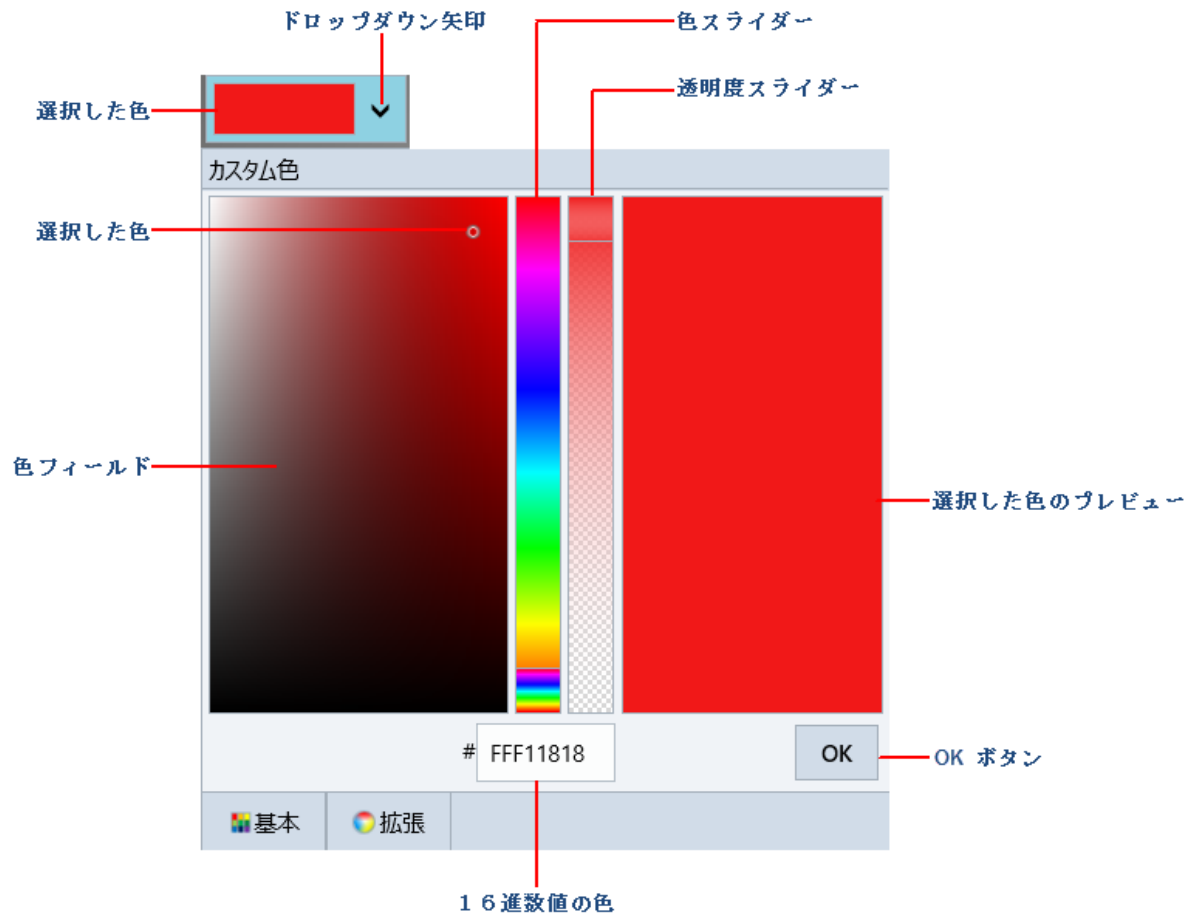


[基本]タブには、以下が含まれます。

- **ドロップダウン矢印**:ドロップダウン矢印をクリックして、C1ColorPicker のウィンドウを開きます。
- **選択された色**:現在選択されている色が ColorPicker のウィンドウに表示されます。
- **ピックした色**:現在ピックされている色は境界線が強調表示されてカラーリストに表示されます。
- **パレット色**:パレット色には、現在選択されているカラーパレットが反映されます。**Palette** プロパティを設定することでパレットを選択できます。
- **ヘッダーパレット**:これらの色は、選択されたパレットにある基本色です。パレット色には、これらの基本色の色調を変えた色がリストされています。
- **標準色**:暗い赤レンガ色、赤、オレンジ、黄色、薄緑、緑、空色、青、濃紺、紫の 10 の標準色が並べられています。
- **最近使用した色**:最近使用した色が最大 10 色まで表示されます。**ShowRecentColors** プロパティを **False** に設定することで、最近使用した色を非表示にすることができます。

詳細モード

詳細モードは、[基本]タブの右側に[詳細]タブとして表示されます。このモードでは、1つの色からさまざまな色調を選択することで、カラーパレットをカスタマイズできます。このモードは **RGB 色**もサポートします。これは、**デザインビュー**で C1ColorPicker の **[プロパティ]**ウィンドウに表示されています。**[詳細]**タブは、次の図のように表示されます。

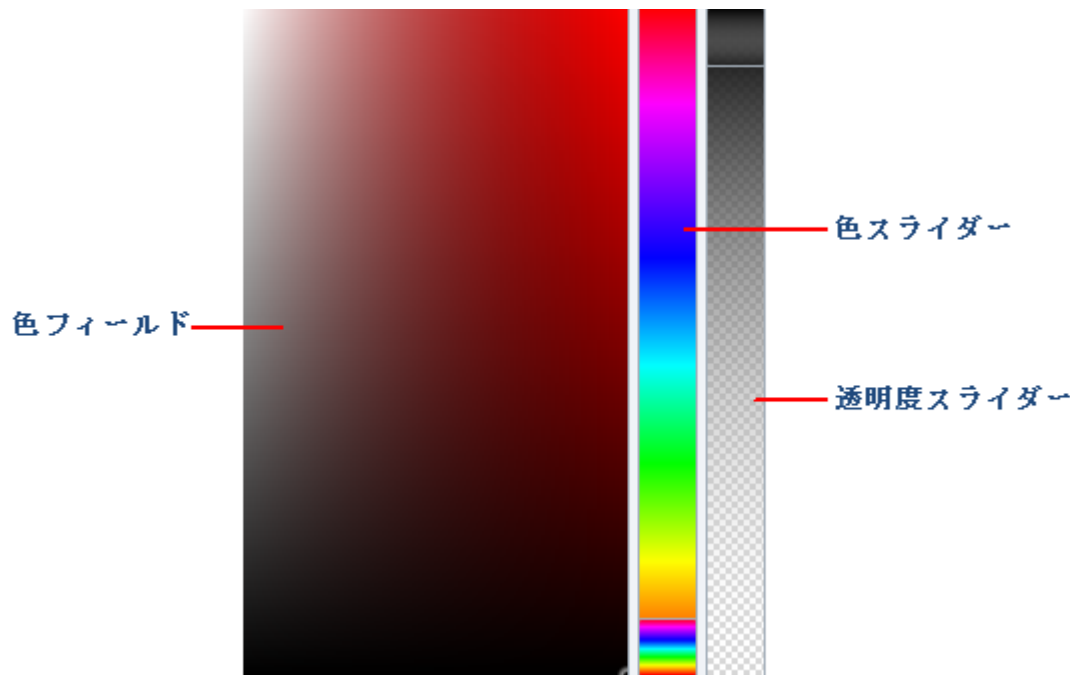


【詳細】タブは、主に以下が含まれます。

- **ドロップダウン矢印**:ドロップダウン矢印をクリックして、C1ColorPicker のウィンドウを開きます。
- **カラーズライダー**:カラーズライダーを使用して、色スペクトルから色を選択できます。**カラーズライダー**を移動して一般色を選択してから、カラーフィールドで選択を調整します。
- **カラーフィールド/ピックした色**:カラーフィールドで色調を選択できます。**ピックした色**は、現在選択されている色を示します。
- **透過スライダー**:このスライダーを使用して、色の透過性を設定できます。不透過に設定することも、部分的または完全な透過に設定することもできます。
- **カラープレビュー**:選択した色をプレビューできます。
- **[OK]ボタン**:色の選択が完了したら、**[OK]**ボタンをクリックして、ドロップダウンを閉じ、その色を選択した色として設定します。

その他のコントロール

C1SpectrumColorPicker: Colorpicker for UWP には、高度な色選択機能を利用できる **C1SpectrumColorPicker** コントロールが含まれます。C1SpectrumColorPicker は次の図のように表示されます。



C1SpectrumColorPicker コントロールには、以下が含まれます。

- **カラーフィールド**: カラーフィールドで色調を選択できます。
- **カラースライダー**: カラースライダーを使用して、色スペクトルから色を選択できます。カラースライダーを移動して一般色を選択してから、カラーフィールドで選択を調整します。
- **透過スライダー**: このスライダーを使用して、色の透過性を設定できます。不透過に設定することも、部分的または完全な透過に設定することもできます。このスライダーは、**ShowAlphaChannel** プロパティが **True** (デフォルト) に設定されている場合にのみ表示されます。

C1HexColorBox: ColorPicker for UWP には、16 進コードで入力するデータを検証するための **C1HexColorBox** コントロールが含まれます。外観は通常のテキストボックスに似ていますが、C1HexColorBox コントロールには 8 文字のコードが表示されます。最初の 2 つの文字は色の透過レベルを表します。たとえば、FF は不透過を表し、00 は透過を表します。残りの 6 文字は、標準の 16 進数色を表します。C1HexColorBox は次の図のように表示されます。

FFC61075

クイックリファレンス

このトピックでは、いくつかの選択プロパティを設定しながら **C1ColorPicker** を作成するための XAML コードについて概要を提供します。ここでは、**DropDownDirection** を AboveOrBelow に設定し、Mode を Basic に設定し、Background を Red に設定します。詳細については、「**C1ColorPicker の使い方**」セクションを参照してください。

XAML

copyCode

```
<Extended:C1ColorPicker x:Name="C1ColorPicker1" HorizontalAlignment="Left" VerticalAlignment="Top"
    Margin="100,73,0,0" Height="77" Width="170" DropDownDirection="AboveOrBelow" Background="Red"
    Mode="Basic"/>
```

クイックスタート

このトピックでは、UWP アプリケーションで **C1ColorPicker** コントロールを追加および設定する方法について説明します。このクイックスタートでは、新しい Visual Studio プロジェクトを作成し、アプリケーションに ColorPicker コントロールを追加し、それらのコントロールの外観をカスタマイズします。また、C1ColorPicker を標準の Rectangle コントロールに追加して、C1ColorPicker の選択プロパティと機能を確認します。C1ColorPicker は、Rectangle に適用されたグラデーションを制御しま

す。実行時に色を選択することで、四角形のグラデーションが変化します。

手順1: C1ColorPicker アプリケーションの作成

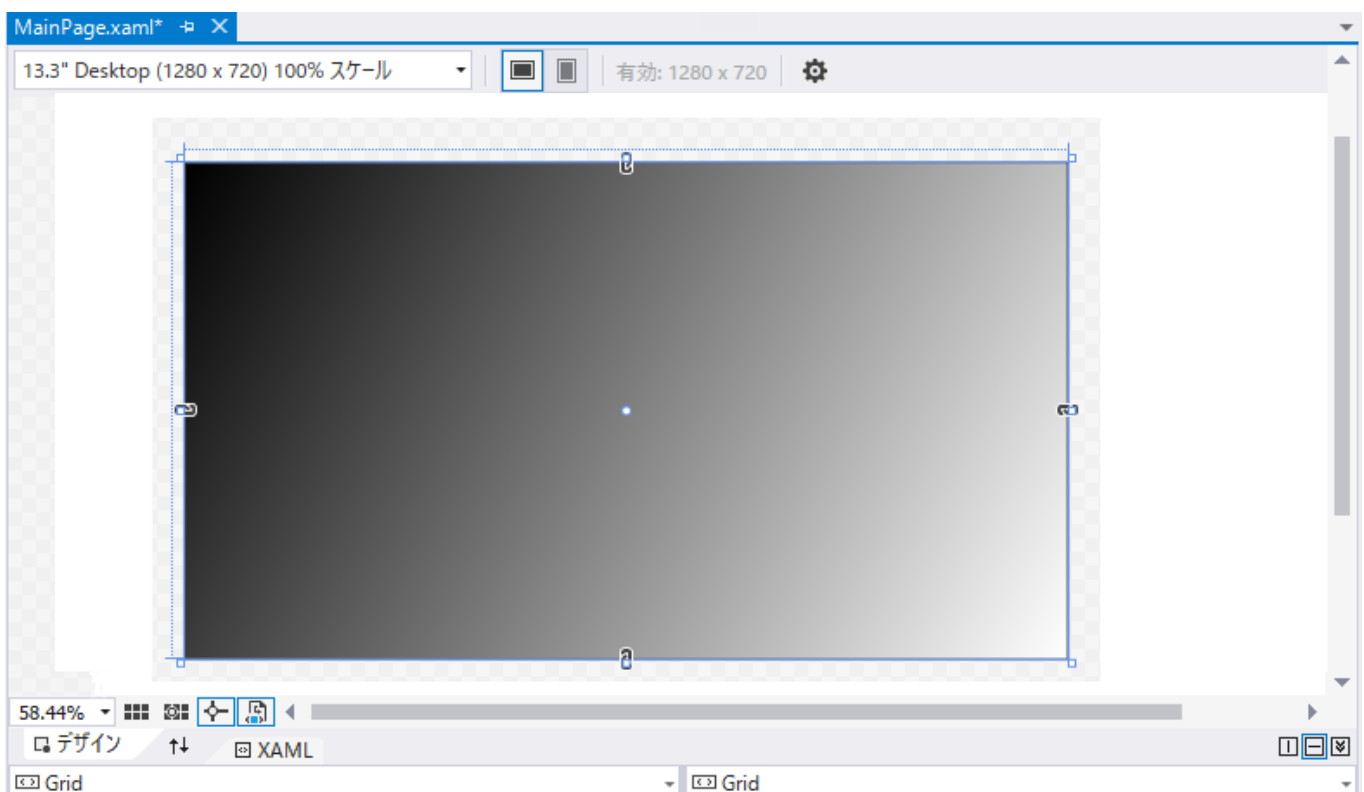
この手順では、新しい UWP プロジェクトを作成し、標準の **Rectangle** コントロールを追加します。また、追加した Rectangle コントロールにグラデーションを適用して外観をカスタマイズします。

1. Visual Studio を開き、[ファイル]→[新規作成]→[プロジェクト]を選択します。
2. [新しいプロジェクト]ダイアログボックスの左ペインで、言語の 1 つを展開します。
 - 選択した言語の下で、[Windows ストア]を選択します。
 - テンプレートリストで、[新しいアプリケーション(XAML)]を選択します。
 - 名前を入力し、[OK]をクリックしてプロジェクトを作成します。
3. デザインビューに切り替え、ツールボックスに移動し、**Rectangle** アイコンをダブルクリックして標準の Rectangle コントロールをグリッドに追加します。
4. ウィンドウのサイズを変更して、四角形をウィンドウ全体に拡大します。
5. 再度 XAML ビューに切り替え、<Rectangle> タグを次のコードに置き換えて塗りつぶしを追加します。

XAML

```
<Rectangle x:Name="Rectangle1" Stroke="Black">
  <Rectangle.Fill>
    <LinearGradientBrush x:Name="colors">
      <GradientStop x:Name="col1" Color="Black" Offset="0" />
      <GradientStop x:Name="col2" Color="White" Offset="1" />
    </LinearGradientBrush>
  </Rectangle.Fill>
</Rectangle>
```

これで、白黒の直線グラデーション塗りつぶしが四角形に追加されます。ページのデザインビューは、次の図のようになります。



これで、UWP アプリケーションを作成し、標準の **Rectangle** コントロールを追加して、外観をカスタマイズできました。次の手順では、C1ColorPicker コントロールをアプリケーションに追加してカスタマイズします。

手順2: C1ColorPicker コントロールの追加

この手順では、引き続き、手順 1 で作成した UWP アプリケーションに **C1ColorPicker** コントロールを追加します。既存の Rectangle コントロールのグラデーション塗りつぶしを制御するために、2 つの C1ColorPicker コントロールを追加します。

1. **デザインビュー**で、四角形を選択し、Visual Studio ツールボックスに移動します。
2. ツールボックスで、**C1ColorPicker** アイコンを見つけて 2 回ダブルクリックし、四角形に 2 つの ColorPicker コントロールを追加します。
3. 追加した **C1ColorPicker** コントロールのサイズを変更し、それらを四角形の中央に配置します。
4. **デザインビュー**で、最初の ColorPicker コントロール (C1ColorPicker1) をクリックし、**[プロパティ]** ウィンドウに移動して、そのプロパティを次のように設定します。
 - **DropDownDirection** プロパティを **AboveOrBelow** に設定して、ColorPicker を開く方向を制御します。
 - **Mode** プロパティを **Advanced** に設定して、高度な色オプションを開きます。
 - **SelectedColor** プロパティを **Black** ("FF000000") に設定します。
5. これらのプロパティを設定すると、XAML ビューは次のようになります。

```
XAML copyCode  
  
<Extended:C1ColorPicker x:Name="C1ColorPicker1" HorizontalAlignment="Left" VerticalAlignment="Top"  
Margin="497,319,0,0" Width="125" Height="67" DropDownDirection="AboveOrBelow" Mode="Advanced"/>
```

6. 2 つめの ColorPicker コントロール (C1ColorPicker2) をクリックし、その SelectedColor プロパティを **White** に設定し、他のプロパティはデフォルトのままにします。ページのデザインビューは次のように表示されます。



これで、UWP アプリケーションのユーザーインターフェースの設計は完了です。しかし、アプリケーションを実行しても、出力ウィンドウに 2 つの C1ColorPicker コントロールが表示されるだけです。これらのコントロールで色を選択しても何も実行されません。次の手順では、追加した ColorPicker コントロールに機能を提供するためのコードをアプリケーションに追加します。

手順3: アプリケーションにコードの追加

この手順では、追加した **C1ColorPicker** コントロールに機能を提供するためのコードを UWP アプリケーションに追加します。既に前の手順でユーザーインターフェースを設計したので、次の手順を実行して機能を追加します。

1. **デザインビュー**で、C1ControlPicker1 を 1 回クリックして選択し、**[プロパティ]** ウィンドウに移動します。
2. **[プロパティ]** ウィンドウで、**[イベント]** アイコンを選択し、**SelectedColorChanged** イベントを見つけ、テキスト領域をダブルクリックします。
3. これで、選択したコントロールのコードビュー (MainPage.xaml.cs) が開き、イベントハンドラが **C1ColorPicker1_SelectedColorChanged** として作成されます。
4. コードの先頭に次の import 文が追加されていることを確認します。

```
Visual Basic copyCode  
  
Imports C1.Xaml  
Imports C1.Xaml.Extended
```

C#	copyCode
<pre>using C1.Xaml; using C1.Xaml.Extended;</pre>	

5. グラデーション値を更新し、**C1ColorPicker1** の SelectedColorChanged イベントハンドラをサブスクライブするには、コードビューで MainPage のコンストラクタの直後に次のコードを追加します (MainPage.xaml.cs)。

Visual Basic	copyCode
<pre>Private Sub UpdateGradient() If C1ColorPicker1 IsNot Nothing And C1ColorPicker2 IsNot Nothing Then Me.col1.Color = Me.C1ColorPicker1.SelectedColor Me.col2.Color = Me.C1ColorPicker2.SelectedColor End If End Sub Private Sub C1ColorPicker1_SelectedColorChanged(sender As Object, e As PropertyChangedEventArgs(Of Windows.UI.Color)) Handles C1ColorPicker1.SelectedColorChanged UpdateGradient() End Sub</pre>	

C#	copyCode
<pre>void UpdateGradient() { if (C1ColorPicker1 != null & C1ColorPicker2 != null) { this.col1.Color = this.C1ColorPicker1.SelectedColor; this.col2.Color = this.C1ColorPicker2.SelectedColor; } } private void C1ColorPicker1_SelectedColorChanged(object sender, C1.Xaml.PropertyChangedEventArgs<Windows.UI.Color> e) { UpdateGradient(); }</pre>	

6. 手順 1、2、および 3 を繰り返して、2 つめの ColorPicker コントロール (C1ColorPicker2) に SelectedColorChanged イベントを追加してサブスクライブします。イベントが作成されたら、次のようにコードでグラデーション値を更新します。

Visual Basic	copyCode
<pre>Private Sub C1ColorPicker2_SelectedColorChanged(sender As Object, e As PropertyChangedEventArgs(Of Windows.UI.Color)) Handles C1ColorPicker2.SelectedColorChanged UpdateGradient() End Sub</pre>	

C#	copyCode

```
private void C1ColorPicker2_SelectedColorChanged(object sender,
PropertyChangedEventArgs<Windows.UI.Color> e)
{
    UpdateGradient();
}
```

これで、UWP アプリケーションにコードを追加して、追加した ColorPicker コントロールに機能を提供できました。次の手順では、このアプリケーションを実行し、実行時のコントロールの機能を確認します。

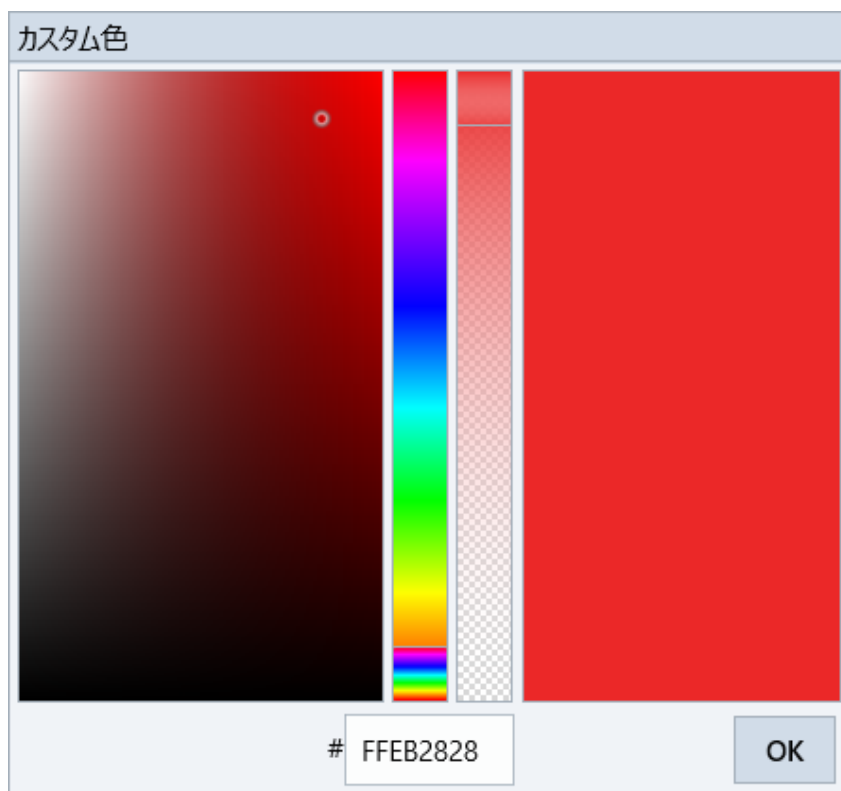
手順4: アプリケーションの実行

UWP アプリケーションを作成し、**C1ColorPicker** コントロールを追加して外観と動作をカスタマイズしたので、このアプリケーションを実行して結果を確認します。

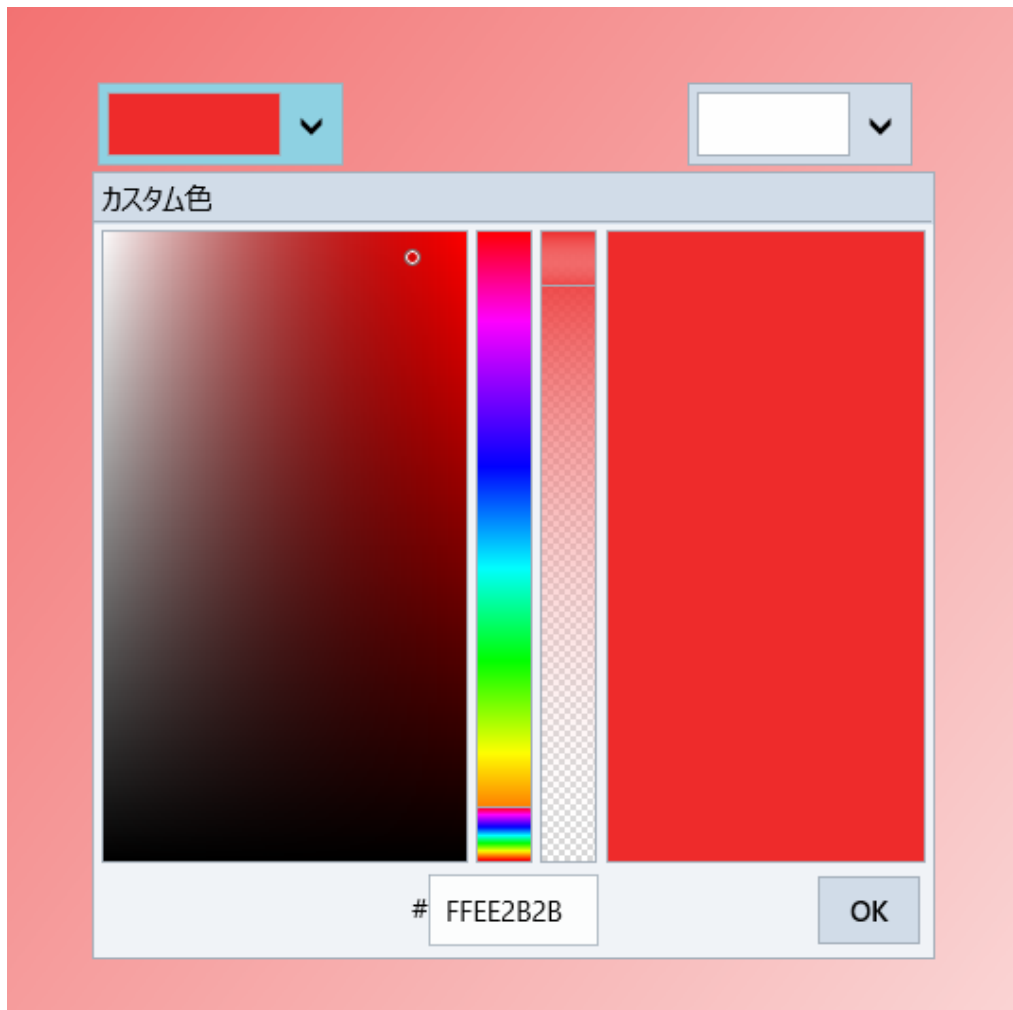
1. **[デバッグ]**メニューで、**[デバッグ開始]**オプションを選択して、出力を表示します。次のように、白黒のウィンドウが表示され、2つの C1ColorPicker コントロールが画面の中央に配置されます。



2. 左側に表示されている ColorPicker のドロップダウン矢印をクリックすると、ドロップダウンボックスが開き、下に詳細モードが表示されます。これは、**手順 2** でこのコントロールに行った変更を反映しています。



3. さまざまな方法で色を選択できます(ここでは赤)。[OK]をクリックします。設定を反映して、選択された色と四角形のグラデーションが変化することを確認します。



4. 右側に表示されている ColorPicker コントロール(C1ColorPicker2)のドロップダウン矢印をクリックして、ドロップダウンボックスがデフォルトの **Basic** モードで開かれ、**[カラーパレット]**、**[標準色]**、**[最近使用した色]**などのタブが表示されることを確認します。



5. ドロップダウンボックスで1つの色を選択します(ここでは黄色)。選択を反映して四角形のグラデーションが変化し、次のように表示されることを確認します。



これで、「クイックスタート」は完了です。ここでは、単純なアプリケーションを作成し、C1ColorPicker コントロールを追加して外観をカスタマイズし、実行時に主な機能をいくつか確認しました。

C1ColorPicker の使い方

このセクションでは、新しい **ColorPicker for UWP** に関連する主要な機能、プロパティなどの重要な項目について説明します。

パレットの設定

ColorPicker for UWP には、Microsoft Office スイートで使用されるテーマに合わせた 20 の定義済みカラーパレットが付属しています。カラーパレットを変更するには、次の手順を実行して、UWP の **C1ColorPicker** コントロールの **Palette** プロパティを設定します。

1. **デザインビュー**で、ツールボックスから標準の **Button** コントロールを追加し、その **Content** プロパティを "**Change Palette**" に設定します。
2. **デザインビュー**に再度切り替え、**Button** コントロールをダブルクリックします。これで、コードビューが開かれ、**Button_Click** イベントハンドラが作成されます。
3. **Button_Click** イベントハンドラに次のコードを追加します。

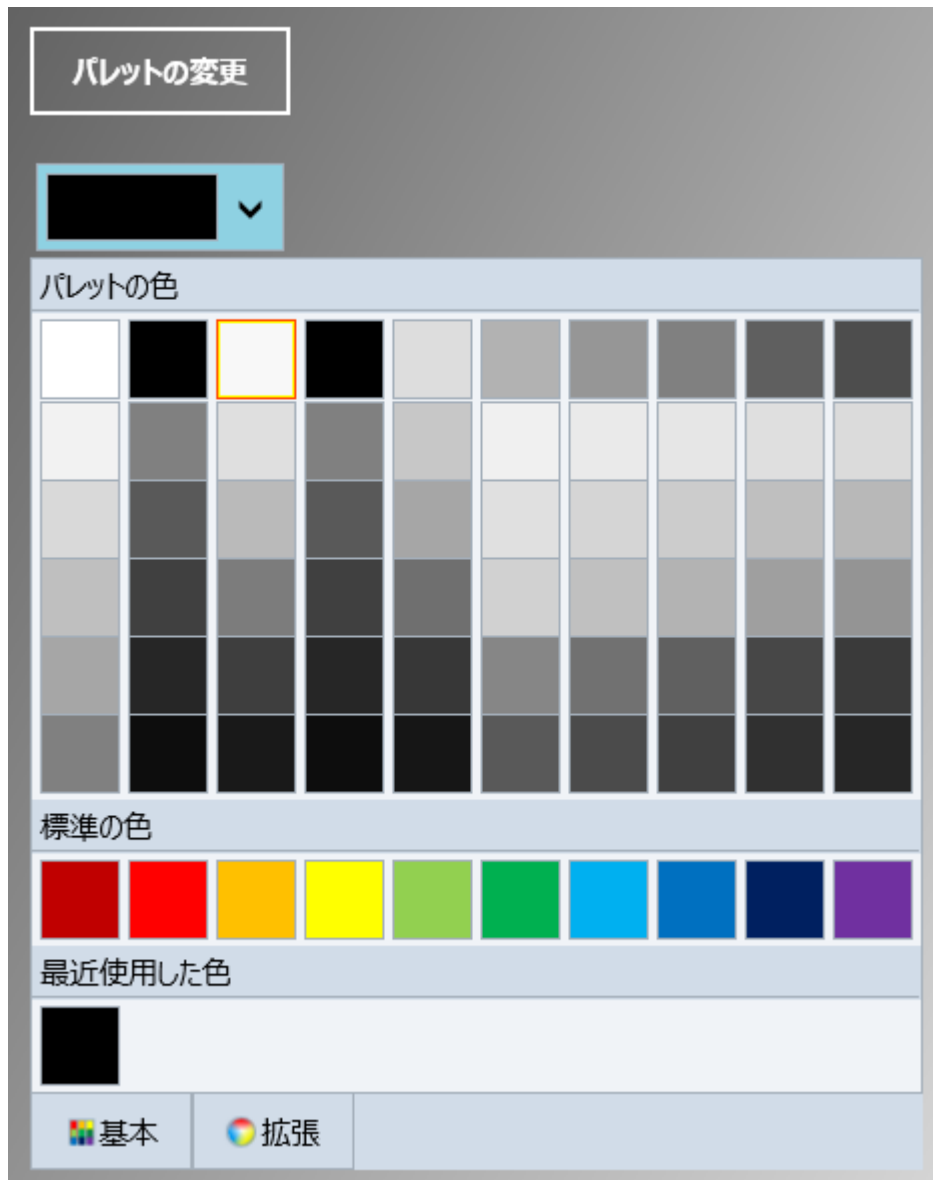
Visual Basic

```
Private Sub Button1_Click(sender As Object, e As RoutedEventArgs) Handles Button1.Click
    Me.C1ColorPicker2.Palette =
    ColorPalette.GetColorPalette(Office2007ColorTheme.GrayScale)
End Sub
```

C#

```
private void Button1_Click(object sender, RoutedEventArgs e)
{
    this.C1ColorPicker2.Palette =
    ColorPalette.GetColorPalette(Office2007ColorTheme.GrayScale);
}
```

4. アプリケーションを実行し、カラーパレットの変更を確認するには、**[デバッグ]**メニューを選択し、**[デバッグ開始]**をクリックします。
5. 実行時環境で、ドロップダウンをクリックして、標準の色オプションが用意されたデフォルトのパレットが表示されることを確認します。
6. 次に、**C1Colorpicker** の上に表示されている **Button** をクリックすると、コードで実装した変更を反映してグレースケールパレットが表示されることを確認します。



パレットのカスタマイズ

アプリケーションで ColorPicker に定義されているカラーパレットを使用することに飽きていませんか。コードに数行を追加するだけで、**ColorPicker for UWP** のカラーパレットをカスタマイズできます。次の手順に従って、カスタムパレットを作成し、ワンクリックで **C1ColorPicker** にパレットを適用できます。

1. ツールボックスに移動し、**Button** アイコンを見つけ、それをダブルクリックしてプロジェクトの**デザイン**ビューに追加します。
2. ボタンのサイズを変更し、**デザイン**ビューに既に追加した C1ColorPicker コントロールの上に配置します。
3. **[プロパティ]**ウィンドウに移動し、ボタンの **Content** プロパティを "**Change Palette**" に設定します。
4. **デザイン**ビューでボタンをダブルクリックし、コードビューに切り替えます。これで、コードに **Button_Click** イベントハンドラが自動的に作成されます。
5. 次の Imports 文をコードの上に追加します。既に追加されている場合は無視してください。

Visual Basic	copyCode
Imports C1.Xaml Imports C1.Xaml.Extended Imports Windows.UI	

C#	copyCode

```
using C1.Xaml;  
using C1.Xaml.Extended;  
using Windows.UI;
```

6. 次のコードを **Button_Click** イベントに追加して、パレットをカスタマイズします。

Visual Basic

copyCode

```
Private Sub Button1_Click(sender As Object, e As RoutedEventArgs) Handles Button1.Click  
    Dim cp1 As New ColorPalette("Pittsburgh")  
  
    cp1.Clear()  
  
    cp1.Add(Color.FromArgb(255, 0, 0, 0))  
  
    cp1.Add(Color.FromArgb(255, 99, 107, 112))  
  
    cp1.Add(Color.FromArgb(255, 255, 255, 255))  
  
    cp1.Add(Color.FromArgb(255, 247, 181, 18))  
  
    cp1.Add(Color.FromArgb(255, 253, 200, 47))  
  
    cp1.Add(Color.FromArgb(255, 43, 41, 38))  
  
    cp1.Add(Color.FromArgb(255, 149, 123, 77))  
  
    cp1.Add(Color.FromArgb(255, 209, 201, 157))  
  
    cp1.Add(Color.FromArgb(255, 0, 33, 71))  
  
    cp1.Add(Color.FromArgb(255, 99, 177, 229))  
  
    C1ColorPicker.Palette = cp1  
End Sub
```

C#

copyCode

```
private void Button1_Click(object sender, RoutedEventArgs e)  
{  
    ColorPalette cp1 = new ColorPalette("Pittsburgh");  
  
    cp1.Clear();  
  
    cp1.Add(Color.FromArgb(255, 0, 0, 0));  
  
    cp1.Add(Color.FromArgb(255, 99, 107, 112));  
  
    cp1.Add(Color.FromArgb(255, 255, 255, 255));  
  
    cp1.Add(Color.FromArgb(255, 247, 181, 18));  
  
    cp1.Add(Color.FromArgb(255, 253, 200, 47));  
  
    cp1.Add(Color.FromArgb(255, 43, 41, 38));  
  
    cp1.Add(Color.FromArgb(255, 149, 123, 77));  
  
    cp1.Add(Color.FromArgb(255, 209, 201, 157));  
}
```

```

cp1.Add(Color.FromArgb(255, 0, 33, 71));

cp1.Add(Color.FromArgb(255, 99, 177, 229));

C1ColorPicker1.Palette = cp1;
}

```

7. アプリケーションを実行します。C1ColorPicker のドロップダウン矢印をクリックして、デフォルトのパレットが表示されることを確認します。
8. 次に、**[Change Palette]**ボタンをクリックし、もう一度 C1ColorPicker のドロップダウン矢印をクリックします。カスタムカラーパレットが表示されることを確認します。



背景色の設定

C1ColorPicker の `Background` プロパティを使用することで、コントロールの背景色を取得または設定できます。UWP アプリケーションでは、設計時または XAML ビューのいずれかで背景色を実装または変更できます。

設計時の場合

設計時に C1ColorPicker の背景色を実装するには、次の手順を実行します。

1. **デザインビュー**で 1 回クリックし、C1ColorPicker コントロールを選択して、**[プロパティ]**ウィンドウに移動します。

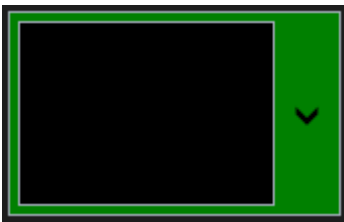
2. **Background** プロパティを見つけ、基底の[エディタ]タブから色 (Green など) を選択します。

XAML の場合

XAML で C1ColorPicker の背景色を実装するには、<Extended:C1ColorPicker> タグに **Background="Green"** を追加します。XAML ビューは次のようになります。

XAML	copyCode
<pre><Extended:C1ColorPicker x:Name="C1ColorPicker1" Margin="283,119,883,0" Height="90" VerticalAlignment="Top" Background="Green"/></pre>	

アプリケーションを実行すると、ColorPicker が緑色で表示されます。



ドロップダウン方向の変更

C1ColorPicker は、ユーザーが実行時にドロップダウン矢印をクリックすると、デフォルトではコントロールの下に表示されます。ただし、設計時、XAML、またはコードを多少変更するだけで、実行時にコントロールが表示される方向を変更できます。

設計時の場合

設計時にドロップダウンウィンドウの方向を変更するには

1. **デザイン**ビューで、C1ColorPicker を 1 回クリックして選択します。
2. [プロパティ]ウィンドウに移動し、**DropDownDirection** プロパティを見つけます。
3. DropDownDirection プロパティの横にあるドロップダウン矢印をクリックし、任意のオプション (**ForceAbove** など) を選択します。

これで、選択したとおりに ColorPicker コントロールの DropDownDirection プロパティが設定されます。

XAML の場合

XAML でドロップダウンウィンドウの方向を変更するには、<Extended:C1ColorPicker> タグに DropDownDirection="ForceAbove" を追加します。XAML ビューは次のようになります。

XAML
<pre><Extended:C1ColorPicker x:Name="C1ColorPicker1" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="147,553,0,0" Height="77" DropDownDirection="ForceAbove"/></pre>

コードの場合

コードでドロップダウンウィンドウが表示される方向をカスタマイズするには、コンストラクタの下に次のコードを追加します。

Visual Basic
<pre>Me.C1ColorPicker1.DropDownDirection = DropDownDirection.ForceAbove</pre>

C#

```
this.C1ColorPicker2.DropDownDirection = DropDownDirection.ForceAbove;
```

アプリケーションを実行します。C1ColorPicker のドロップダウン矢印をクリックすると、次の図のようにドロップダウンが C1ColorPicker コントロールの上に表示されることを確認します。



基本モードから最近使用した色の非表示

Basic モードでは、C1ColorPicker に[最近使用した色]タブが表示され、ColorPicker で選択して最近使用した色のアイコンがこのタブに表示されます。設計時、XAML、またはコードで[最近使用した色]タブを非表示にすることで、C1ColorPicker の外観をカスタマイズできます。

設計時の場合

設計時に[最近使用した色]タブを非表示にするには

1. **デザインビュー**で、C1ColorPicker コントロールを 1 回クリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、**ShowRecentColors** プロパティを見つけます。
3. ShowRecentColors プロパティをオフにして、[最近使用した色]タブを非表示にします。

XAML の場合

XAML で[最近使用した色]タブを非表示にするには、<Extended:C1ColorPicker> タグに ShowRecentColors="False" を追加して ShowRecentColors プロパティを false に設定します。XAML コードは次のようになります。

Extended Library for UWP

XAML

```
<Extended:C1ColorPicker x:Name="C1ColorPicker2" HorizontalAlignment="Left"
    VerticalAlignment="Top" Margin="143,101,0,0" Width="179" Height="84"
    ShowRecentColors="False"/>
```

コードの場合

コードビューで、[最近使用した色]タブを非表示にするには、MainPage コンストラクタに次のコードを追加します。

Visual Basic

```
Me.C1ColorPicker1.ShowRecentColors = False
```

C#

```
this.C1ColorPicker2.ShowRecentColors = false;
```

アプリケーションを実行し、ドロップダウンウィンドウに[最近使用した色]タブが表示されていないことを確認します。

