Basic Library for UWP

2018.04.10 更新

グレープシティ株式会社

Basic Library for UWP

目次

Copyright © GrapeCity, Inc. All rights reserved.

Copyright © GrapeCity, Inc. All rights reserved.

Copyright © GrapeCity, Inc. All rights reserved.

Copyright © GrapeCity, Inc. All rights reserved.

Copyright © GrapeCity, Inc. All rights reserved.

Copyright © GrapeCity, Inc. All rights reserved.

Basic Library for UWP

ComponentOne for UWP 内でデータの視覚化、レイアウト、入力などを行うための UI コントロールを提供します。これらのコントロールは、よく使用されている Silverlight コントロールに基づいて、UWP プラットフォームのユーザーエクスペリエンスを向上させるように設計されています。より魅力的で優れた Windows ストアアプリケーションの構築に役立つ独自の強力な機能を提供します。

Basic Library には、以下のコントロールが含まれます。

- **CollectionView for UWP**
  CollectionView for UWP を使用すると、**ICollectionView** インタフェースを一層強力に実装できます。**C1CollectionView** クラスは、任意のコレクションのソート、フィルタ処理、グループ化、変更などの機能を新たに提供します。Windows ストアアプリケーションで **CollectionViewSource** とまったく同様に **C1CollectionView** を使用して、何も犠牲にすることなくさらに多くの機能をすぐに追加できます。
- **Input for UWP**
  電話番号、郵便番号、パーセンテージなどの入力をスマートに行います。**Input for UWP** は、マスク付き入力用のコントロールと数値入力用のコントロールの2つを提供します。書式設定されたテキストを自動的に表示すると共に、有効な入力をすばやく収集できます。
- **Layout Panels for UWP**
  Layout Panels for UWP を使用して Windows ストアアプリケーションのコンテンツのフローと配置を制御します。**C1WrapPanel** を使用すると、コンテンツを垂直または水平方向に折り返すことができます。**C1DockPanel** を使用すると、パネルの端にコンテンツをドッキングできます。**C1UniformGrid** を使用すると、コンテンツをグリッド内で整然と表示できます。
- **ListBox for UWP**
  連結データをリスト表示する場合は、**ListBox for UWP** に含まれる高パフォーマンスな2つのコントロールを利用してください。**C1ListBox** コントロールや **C1TileListBox** コントロールを使用して、リストをタイル状にレイアウトして表示したり、光学ズーム付きで表示することができます。これらのコントロールは UI の仮想化をサポートしています。そのため、動作は極めて高速であり、数千個の項目を表示してもパフォーマンスはほとんど低下しません。
- **Menu for UWP**
  Menu for UWP を使用すると、Windows ストアアプリケーションに従来の "File" メニューシステムを追加できます。**C1Menu** コントロールは、深くネストされた項目と垂直方向をサポートする従来どおりの外観のメニューを備えた、リアルなデスクトップのようなルックアンドフィールを提供します。
- **RadialMenu for UWP**
  RadialMenu for UWP を使用すると、Windows ストアアプリケーションに魅力的なラジアルメニューシステムを追加できます。**C1RadialMenu** コントロールは、よく使用されているマイクロソフトアプリケーションをモデルにして、従来のコンテキストメニューに代わるユニークなタッチ操作向きのメニューを提供します。
- **TabControl for UWP**
  TabControl for UWP を使用すると、コンテンツをタブとして整理し、それらのタブ間を移動することができます。タブは、空きスペースを有効利用して、選択可能な項目をすべてユーザーに表示するために役立ちます。タブはページの上下左右に配置でき、数種類の形状と組み込み機能がサポートされています。
- **TreeView for UWP**
  TreeView for UWP を使用すると、データ項目を階層的に表示できます。使い慣れたツリービュー UI を Windows 8 アプリケーションで使用できるようになりました。折りたたみ可能なノード、階層テンプレート、チェックボックスノード、編集、ドラッグ &ドロップ操作などがサポートされています。
- **DropDown for UWP**
  DropDown for UWP を使用すると、任意のタイプのカスタム入力用ドロップダウンを作成できます。**C1DropDown** コントロールは、ポップアップやフライアウトにわずらわされることなく、カスタムドロップダウンエディタを作成できるようにします。これを使用して、カラーピッカー、オートコンプリートテキストボックスから、階層型コンボボックスまで、考えられるあらゆる種類の特殊なドロップダウンを作成できます。

CollectionView for UWP

CollectionView for UWP を使用すると、**ICollectionView** インタフェースを一層強力に実装できます。**C1CollectionView** クラスは、任意のコレクションのソート、フィルタ処理、グループ化、変更などの機能を新たに提供します。Windows ストアアプ

Copyright © GrapeCity, Inc. All rights reserved.

リケーションで **CollectionViewSource** とまったく同様に C1CollectionView を使用して、何も犠牲にすることなくさらに多くの機能をすぐに追加できます。

主な特長

CollectionView for UWP には、次の主な特長があります。

- **ソート、フィルタ処理、グループ化**
  C1CollectionView クラスと **IC1CollectionView** インタフェースは、UWP アプリケーションにコレクションのソート、フィルタ処理、およびグループ化のサポートを提供します。オブジェクトモデルと機能は、WPF、Silverlight、Windows Phone で提供されている **ICollectionView** インタフェースとほぼ同じなので、新たな学習は必要ありません。
- **CollectionViewSource より強力**
  UWP の標準の **ICollectionView** インタフェースおよび **CollectionViewSource** の実装は、WPF や Silverlight に比べて機能が制限されています。たとえば、**ICollectionView** の UWP 実装は、ソート、フィルタ処理、編集をサポートしていません。C1CollectionView クラスは、これらのサポートされていない要素を追加して、必要な機能を実行できるようにします。
- **任意のコントロールとの併用**
  C1CollectionView は **ICollectionView** インタフェースを完全に実装しているため、標準の項目コントロールのすべてでデータソースとして使用できます。多くの ComponentOne コントロールは C1CollectionView クラスを内部で自動的に使用しますが、インタフェースはパブリックなので、任意のコントロールと組み合わせて使用することができます。
- **編集可能なライブ行セット**
  C1CollectionView はライブ行セットを提供します。C1CollectionView 内のデータに加えられた変更はすべて、基底のコレクションに自動的に伝えられます。bはまた、WPF や Silverlight と同じ **IEditableCollectionView** インタフェースを追加で実装するため、[Esc]キーを押すと編集中の項目に加えられたすべての変更を取り消すことができる機能など、優れた編集エクスペリエンスを実現できます。
- **使い慣れたオブジェクトモデル**
  IC1CollectionView インタフェースは WPF および Silverlight の **ICollectionView** に基づいているため、コレクションのソート、フィルタ処理、グループ化はおなじみの作業です。また、C1CollectionView クラスは、WPF、Silverlight、および Windows Phone バージョンと互換性があるため、既存のコードを容易に再利用できます。

C1CollectionView の使い方

C1CollectionView は **IC1CollectionView** インタフェースを実装し、このインタフェースは標準の **ICollectionView** インタフェースを実装しています。WPF の標準の **CollectionView** クラスと同様に、**C1CollectionView** は、最新の項目管理、項目選択、ソート、グループ化、フィルタ処理、および編集機能をサポートします。このマニュアル内のトピックで十分な情報が得られない場合は、WPF や Silverlight の **ICollectionView** の使用方法を Web で検索すれば、UWP でも活用できる有益な情報を容易に見つけることができます。

C1CollectionView クラスは **C1.Xaml** アセンブリにあります。

C1CollectionView を使用するには、ビジネスオブジェクトの **IEnumerable** コレクションを使用してインスタンス化します。

Visual Basic コードの書き方

```
Visual Basic
Dim customers As List(Of Customer) = Await GetCustomerData()
Dim view = New C1.Xaml.C1CollectionView(customers)
```

C# コードの書き方

```
C#
List<Customer> customers = await GetCustomerData();
var view = new C1.Xaml.C1CollectionView(customers);
```


Copyright © GrapeCity, Inc. All rights reserved.

次に、**ItemsControl** またはデータグリッドに連結して、C1CollectionView の使用を開始します。

Visual Basic コードの書き方

```
Visual Basic
C1FlexGrid1.ItemsSource = view
```

C# コードの書き方

```
C#
c1FlexGrid1.ItemsSource = view;
```

C1CollectionView と CollectionViewSource

UWP の標準の **ICollectionView** インタフェースおよび **CollectionViewSource** の実装は、WPF や Silverlight に比べて機能が制限されています。たとえば、**ICollectionView** の UWP 実装は、ソート、フィルタ処理、コレクションの編集をサポートしていません。**C1CollectionView** クラスは、これらのサポートされていない要素を追加して、必要な機能を実行できるようにします。**IC1CollectionView** インタフェースは WPF および Silverlight の **ICollectionView** に基づいているため、それらのプラットフォームに精通している場合は、**C1CollectionView** を使用したソート、フィルタ処理、グループ化もおなじみの作業です。**CollectionViewSource** クラスに精通している場合、**C1CollectionView** への移行は極めて簡単です。

基底の **Customer** オブジェクトのリストが同じだとして、**C1FlexGrid** コントロールを **CollectionViewSource** に連結する例と C1CollectionView に連結する例を比べてみます。

CollectionViewSource:

Visual Basic コードの書き方

```
Visual Basic
Dim customers As List(Of Customer) = Await GetCustomerData()
Dim view = New CollectionViewSource()
view.Source = customers
c1FlexGrid1.ItemsSource = view.View
```

C# コードの書き方

```
C#
List<Customer> customers = await GetCustomerData();
var view = new CollectionViewSource();
view.Source = customers;
c1FlexGrid1.ItemsSource = view.View;
```

C1CollectionView:

Visual Basic コードの書き方

```
Visual Basic
Dim customers As List(Of Customer) = Await GetCustomerData()
Dim view = New C1.Xaml.C1CollectionView(customers)
c1FlexGrid1.ItemsSource = view
```


Copyright © GrapeCity, Inc. All rights reserved.

C# コードの書き方

```
C#
List<Customer> customers = await GetCustomerData();
var view = new C1.Xaml.C1CollectionView(customers);
c1FlexGrid1.ItemsSource = view;
```

MVVM で作業している場合は、ビューモデルで **IC1CollectionView** 型のプロパティを公開し、ビューモデル内でコレクションの内容を設定するだけで済みます。その後、XAML で、そのプロパティに **C1FlexGrid**（または使用している任意のコントロール）の **ItemsSource** プロパティを連結します。

Visual Basic コードの書き方

```
Visual Basic
''' <summary>
''' 顧客のコレクションを取得します。
''' </summary>
Public ReadOnly Property Customers() As C1.Xaml.IC1CollectionView
Get
    Return view
End Get
End Property
```

C# コードの書き方

```
C#
/// <summary>
/// 顧客のコレクションを取得します。
/// </summary>
public C1.Xaml.IC1CollectionView Customers
{
  get { return view; }
}
```

コントロールをソートする

他のプラットフォームで **CollectionView** 実装を使用する場合と同様に、**C1CollectionView** を使用してコレクション内の項目をソートできます。**C1CollectionView** をソートしても、基底のデータセットは影響を受けません。

たとえば、SortDescription オブジェクトを渡す **SortDescriptions** プロパティを使用して、項目をソートできます。

Visual Basic コードの書き方

```
Visual Basic
Dim list = New System.Collections.ObjectModel.ObservableCollection(Of Customer)()
' リストから C1CollectionView を作成します
Dim _view As New C1.Xaml.C1CollectionView(list)
' 顧客を国に基づいてソートします
_view.SortDescriptions.Add(New C1.Xaml.SortDescription("Country", C1.Xaml.ListSortDirection.Ascending))
```

C# コードの書き方

Copyright © GrapeCity, Inc. All rights reserved.

```
C#
var list = new System.Collections.ObjectModel.ObservableCollection<Customer>();
// リストから C1CollectionView を作成します
C1.Xaml.C1CollectionView _view = new C1.Xaml.C1CollectionView(list);
// 顧客を国に基づいてソートします
_view.SortDescriptions.Add(new C1.Xaml.SortDescription("Country", C1.Xaml.ListSortDirection.Ascending));
```

ここで、"Country" はソートの基準にするプロパティの名前です。**ListSortDirection** パラメータに基づいて昇順（A-Z）または降順（Z-A）でソートできます。

複数のプロパティまたは列でソートする方法

SortDescriptions を追加するだけで、複数のプロパティに基づくソートが可能です。複数のプロパティに基づいてソートする場合は、**DeferRefresh** メソッドを使用して各ソート後の自動リフレッシュを遅延させ、各ソートが一度だけ適用されるようにする必要があります。

Visual Basic コードの書き方

```
Visual Basic
' リフレッシュが一度ですむように DeferRefresh を使用して複数のプロパティをソートします
Using _view.DeferRefresh()
  _view.SortDescriptions.Clear()
  _view.SortDescriptions.Add(New C1.Xaml.SortDescription("Country", C1.Xaml.ListSortDirection.Ascending))
  _view.SortDescriptions.Add(New C1.Xaml.SortDescription("Name", C1.Xaml.ListSortDirection.Ascending))
End Using
```

C#コードの書き方

```
C#
// リフレッシュが一度ですむように DeferRefresh を使用して複数のプロパティをソートします
using (_view.DeferRefresh())
{
   _view.SortDescriptions.Clear();
   _view.SortDescriptions.Add(new C1.Xaml.SortDescription("Country", C1.Xaml.ListSortDirection.Ascending));
   _view.SortDescriptions.Add(new C1.Xaml.SortDescription("Name", C1.Xaml.ListSortDirection.Ascending));
}
```

メモ: コレクションが **INotifyCollectionChanged** を実装している場合は、ソートの設定後であっても、データに加えられた変更がすべてソートに適用されます。

パフォーマンスを向上できる高度なソートについては、**CustomSort** プロパティを参照してください。

コントロールのフィルタ処理

C1CollectionView を使用すると、コレクションをフィルタ処理して、元のコレクションのうち、指定された述語が true を返す要素のみを含む新しいサブセットを作成できます。**C1CollectionView** をフィルタ処理しても、基底のデータセットは影響を受けません。**Filter** プロパティは、項目をビューに入れるかどうかを判断するために使用されるコールバックを取得または設定します。

たとえば、Filter プロパティに述語を設定すると、その述語でフィルタ処理されたリストが作成されます。

Visual Basic コードの書き方

Copyright © GrapeCity, Inc. All rights reserved.

```
Visual Basic
' 顧客の監視可能なリストを作成します
Dim list = New System.Collections.ObjectModel.ObservableCollection(Of Customer)()
' リストから C1CollectionView を作成します
_view = New C1.Xaml.C1CollectionView(list)
' 国（オーストリア）でフィルタ処理します。オーストリア以外の顧客は除外されます。
_view.Filter = Sub(item As Object)
   Dim c As Customer = TryCast(item, Customer)
   If c IsNot Nothing Then
       If c.Country.Equals("Austria") Then
           Return True
   End If
    End If
    Return False
    End Sub
```

C# コードの書き方

```
C#
// 顧客の監視可能なリストを作成します
var list = new System.Collections.ObjectModel.ObservableCollection<Customer>();
// リストから C1CollectionView を作成します
_view = new C1.Xaml.C1CollectionView(list);
// 国（オーストリア）でフィルタ処理します。オーストリア以外の顧客は除外されます。
_view.Filter = delegate(object item)
{
   Customer c = item as Customer;
   if (c != null)
   {
       if (c.Country.Equals("Austria"))
           return true;
   }
   return false;
};
```

メモ: コレクションが **INotifyCollectionChanged** を実装している場合は、フィルタ処理の設定後であっても、データに加えられた変更がすべてフィルタ処理に適用されます。

コントロールのグループ化

C1CollectionView では、**GroupDescriptions** プロパティを使用してコレクションを特定のプロパティでグループ化できます。

たとえば、コレクションを "Country" プロパティでグループ化するには次のようにします。

Visual Basic コードの書き方

```
Visual Basic
Dim list = New System.Collections.ObjectModel.ObservableCollection(Of Customer)()
' リストから C1CollectionView を作成します
_view = New C1.Xaml.C1CollectionView(list)
'  顧客を国別にグループ化します
_view.GroupDescriptions.Add(New C1.Xaml.PropertyGroupDescription("Country"))
```


Copyright © GrapeCity, Inc. All rights reserved.

C#コードの書き方

```
C#
var list = new System.Collections.ObjectModel.ObservableCollection<Customer>();
// リストから C1CollectionView を作成します
_view = new C1.Xaml.C1CollectionView(list);
// 顧客を国別にグループ化します
_view.GroupDescriptions.Add(new C1.Xaml.PropertyGroupDescription("Country"));
```

> **メモ**: コレクションが **INotifyCollectionChanged** を実装している場合は、グループ化の設定後であっても、データに加えられた変更がすべてグループ化に適用されます。

C1FlexGrid コントロールは、**C1CollectionView** に基づくグループ化をサポートします。他のコントロールを使用してグループ化を行う場合は、各グループの外観を制御する **GroupStyle** を定義する必要があります。

次に、標準の **ListBox** コントロールに対して定義された **GroupStyle** の例を示します。

```
<ListBox
   Name="_listBox"
   ItemsSource="{Binding Customers}">
   <ListBox.GroupStyle>
      <GroupStyle>
         <GroupStyle.HeaderTemplate>
            <DataTemplate>
               <TextBlock Text="{Binding}" />
            </DataTemplate>
         </GroupStyle.HeaderTemplate>
      </GroupStyle>
   </ListBox.GroupStyle>
</ListBox>
```

カスタムグループ化

数値でグループ化する例を考えてみます。少数の切りのいい整数のリストがない場合は、一意の値を集めると、管理不能な大きさになることがあります。このような場合は、代わりに、「0 ～ 100」、「5000 以上」などのいくつかの範囲に項目をグループ化するカスタムグループ化アクションを適用できます。それには、**IValueConverter** を **PropertyGroupDescription** パラメータに渡してカスタムグループ化を実行します。

たとえば、次のコードは、各グループをアルファベット1文字（Countries: A、Countries: B、Countries: C など）としてリストして、顧客コレクションを国に基づいてグループ化します。Countries: A グループには、アルジェリア、アルゼンチン、オーストリアに属するすべての項目が含まれます。前述のコードスニペットを次のように変更します

Visual Basic コードの書き方

```
Visual Basic
_view.GroupDescriptions.Add(New C1.Xaml.PropertyGroupDescription("Country", New
GroupByCountryAtoZConverter()))
```

C#コードの書き方

```
C#
```


Copyright © GrapeCity, Inc. All rights reserved.

```
_view.GroupDescriptions.Add(new C1.Xaml.PropertyGroupDescription("Country", new
GroupByCountryAtoZConverter()));
```

さらに、次の **GroupByCountryAtoZConverter** クラスをプロジェクトに追加します。

Visual Basic コードの書き方

Visual Basic

```
Public Class GroupByCountryAtoZConverter
Implements IValueConverter
Public Function Convert(value As Object, targetType As Type, parameter As Object, culture As String) As Object
        If value IsNot Nothing Then
            Return value.ToString()(0)
        End If
        Return "Undefined"
End Function
Public Function ConvertBack(value As Object, targetType As Type, parameter As Object, culture As String) As Object
        Throw New NotImplementedException()
End Function
End Class
```

C# コードの書き方

C#

```
public class GroupByCountryAtoZConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, string culture)
    {
        if (value != null)
        {
            return value.ToString()[0];
        }
        return "Undefined";
    }
    public object ConvertBack(object value, Type targetType, object parameter, string culture)
    {
        throw new NotImplementedException();
    }
}
```

DropDown for UWP

DropDown for UWP を使用して、任意のタイプのカスタム入力用ドロップダウンを作成できます。**C1DropDown** コントロールは、ポップアップやフライアウトにわずらわされることなく、カスタムドロップダウンエディタを作成できるようにします。これを使用して、カラーピッカー、オートコンプリートテキストボックスから、階層型コンボボックスまで、考えられるあらゆる種類の特殊なドロップダウンを作成できます。

主な特長

DropDown for UWP には、次の主要な機能があります。

Copyright © GrapeCity, Inc. All rights reserved.

- **ドロップダウンのコンテンツとして任意の UI を格納**
  C1DropDown コントロールを使用すると、特殊なドロップダウンエディタの作成を簡単に、しかも完全に制御できます。自由に独自のドロップダウンコンテンツを設計し、ヘッダー部分に表示する値を設定します。たとえば、ドロップダウン部分に **C1TreeView** を配置し、選択されたノードをヘッダーに表示するような階層型コンボボックスを作成できます。
- **開閉範囲の自動検出**
  ドロップダウン方向をヘッダー部分の上または下に設定できます。ページに十分な余地がない場合、C1DropDown コントロールは自動的に反対方向に表示されます。AutoClose プロパティを使用して、コントロールがフォーカスを失ったときにドロップダウンを自動的に閉じるかどうかを指定できます。

クイックスタート

このクイックスタートガイドは、**C1DropDown** コントロール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ガイドでは、最初に Visual Studio で新しいプロジェクトを作成し、アプリケーションに C1DropDown 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: C1DropDown コントロールによるア プリケーションの作成

この手順では、Visual Studio で、**DropDown for UWP** を使用して UWP アプリケーションを作成します。

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、言語の下で[**Windows ストア**]を選択し、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. **MainPage.xaml** をまだ開いていない場合は開きます。
4. 開始タグ `<Page>` を編集して、次の名前空間を追加します。

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

5. ページ内の `<Grid> </Grid>` タグを見つけ、次のマークアップを挿入して Grid の行を定義します。

```
<Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition/>
    </Grid.RowDefinitions>
```

6. 行定義のすぐ下に、次のマークアップを使用して CheckBox を追加します。

```
<CheckBox Content="AutoClose - determines if C1DropDown will automatically close when focus is lost."
IsChecked="{Binding ElementName=c1DropDown1, Path=AutoClose, Mode=TwoWay}" Margin="10"/>
```

7. 次のマークアップを使用して、**C1DropDownButton** コントロールを追加します。

```
<Xaml:C1DropDownButton Grid.Row="1" Background="White" x:Name="c1DropDown1" Padding="2"
Width="150" HorizontalAlignment="Center" VerticalAlignment="Center" > </Xaml:C1DropDownButton>
```

8. 次に、Header のコンテンツを追加して、**C1DropDownButton.Header** をカスタマイズします。

```
<Xaml:C1DropDownButton.Header>
  <Border x:Name="dropDownBorder" Background="White" />
</Xaml:C1DropDownButton.Header>
```


Copyright © GrapeCity, Inc. All rights reserved.

ここまでの成果

この手順では、新しい Visual Studio 2013 Windows ストアアプリケーションを追加し、**CheckBox** コントロールを追加し、**C1DropDownBox** コントロールを追加してカスタマイズを開始しました。

手順2：C1DropDown コントロールへのコ ンテンツの追加

この手順では、前の手順で追加した **C1DropDown** コントロールにコンテンツを追加します。

1. `<Xaml:C1DropDownButton.Header>` のすぐ下に次のマークアップを追加します。これで、**C1DropDown** コントロールにコンテンツを追加できます。

```
<Xaml:C1DropDownButton.Content>
</Xaml:C1DropDownButton.Content>
```

2. `<Xaml:C1DropDownButton.Content> </Xaml:C1DropDown.Content>` タグの間にカーソルを置きます。
3. Visual Studio ツールボックスで **C1TileListBox** コントロールを見つけます。このコントロールをダブルクリックしてページに追加します。
4. 次のように、開始タグ `<Xaml:C1TileListBox>` を編集します。

```
<Xaml:C1TileListBox x:Name="colorListBox"
                    Height="180"
                    Orientation="Horizontal"
                    ItemTapped="colorListBox_ItemTapped"
                    SelectionMode="None"
                    BorderBrush="{StaticResource ComboBoxPopupBorderThemeBrush}"
                    BorderThickness="{StaticResource ComboBoxPopupBorderThemeThickness}"
                    Background="{StaticResource ComboBoxPopupBackgroundThemeBrush}">
```

5. `<Xaml:C1TileListBox> </Xaml:C1TileListBox>` タグの間にカーソルを置き、次のマークアップを追加します。これで、実行時に **C1DropDownButton** コントロールの背景色を変更できます。

```
<Border Background="Black" BorderBrush="White" BorderThickness="1"/>
            <Border Background="DarkGray"/>
            <Border Background="White" BorderBrush="Black" BorderThickness="1"/>

            <Border Background="DarkBlue" />
            <Border Background="Blue" />
            <Border Background="Cyan" />

            <Border Background="Teal" />
            <Border Background="Green" />
            <Border Background="Lime" />

            <Border Background="SaddleBrown"/>
            <Border Background="Orange" />
            <Border Background="Yellow" />

            <Border Background="Maroon" />
            <Border Background="Red" />
            <Border Background="Magenta" />
```


Copyright © GrapeCity, Inc. All rights reserved.

6. ページを右クリックし、リストから[**コードの表示**]を選択します。次の名前空間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;
```

7. 次のコードを追加して、**colorListBox_ItemTapped** イベントを処理します。

C# コードの書き方

```
C#
private void colorListBox_ItemTapped(object sender, C1TappedEventArgs e)
        {
            C1ListBoxItem item = sender as C1ListBoxItem;
            if (item != null)
            {
                Border b = item.Content as Border;
                if (b != null)
                {
                    dropDownBorder.Background = b.Background;
                }
            }
            c1DropDown1.IsDropDownOpen = false;
        }
```

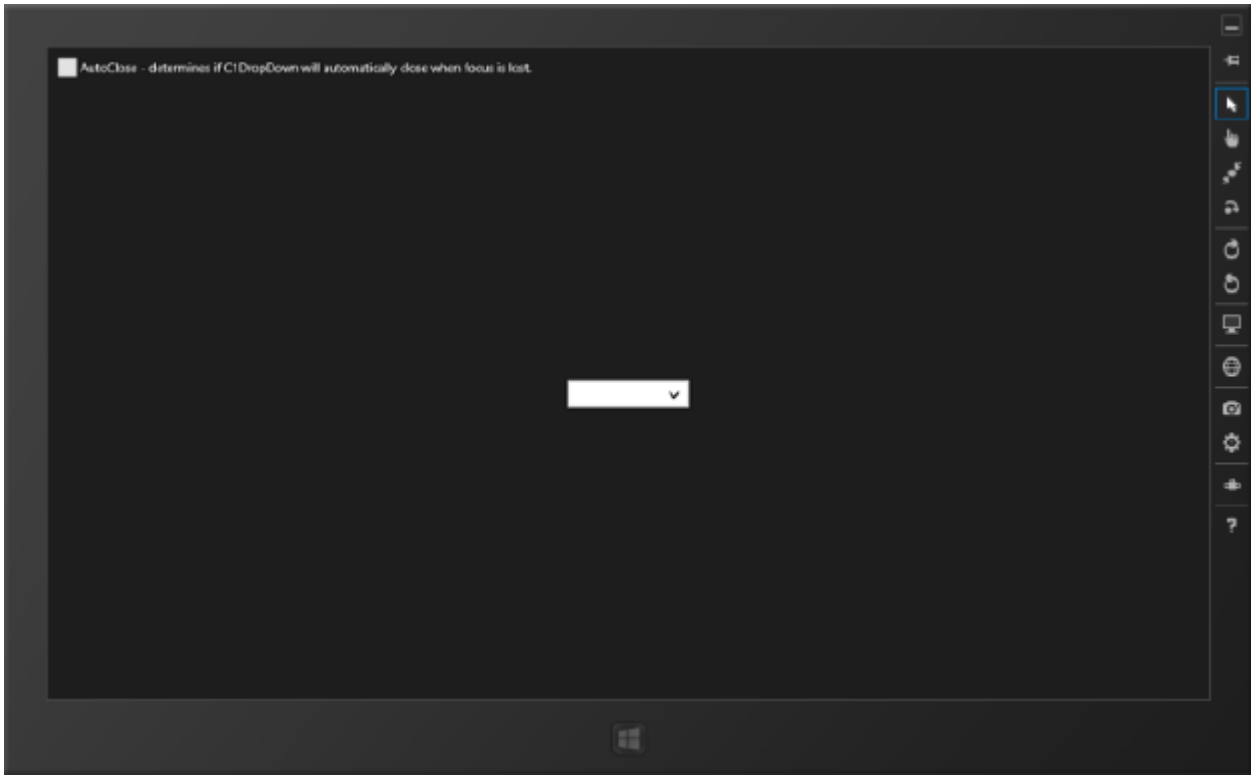
ここまでの成果

この手順では、C1DropDownButton コントロールにコンテンツを追加しました。次の手順では、このアプリケーションを実行します。

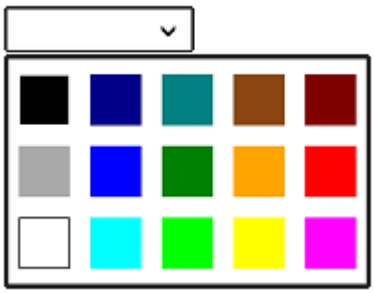
手順3：C1DropDown アプリケーションの 実行

1. [F5]キーを押すか、デバッグを開始して、アプリケーションを実行します。次の図のようになります。

Copyright © GrapeCity, Inc. All rights reserved.



2. ドロップダウンボタンをタップまたはクリックすると、**C1DropDownButton** コントロールは次の図のように表示されます。



3. **C1TileListBox** から色を選択すると、C1DropDownButton は次の図のようになります。

ここまでの成果

このクイックスタートでは、新しい Windows ストアアプリケーションを追加し、そのアプリケーションに C1DropDownButton コントロールを追加し、**C1DropDownButton** のコンテンツ領域に C1TileListBox を追加しました。

C1DropDown の使い方

DropDown for UWP には **C1DropDown** 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ドロップダウンボックスコントロールです。コントロールや画像などを対話式の入力ボックスに追加できます。XAML ウィンドウに追加された C1DropDown コントロールは、完全に機能する入力コントロールになります。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。

基本的なイベント

Copyright © GrapeCity, Inc. All rights reserved.

DropDown 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、**C1DropDown** コントロールをカスタマイズできます。

イベント	説明
IsDropDownOpenChanged	IsDropDownOpen プロパティが変更されたときに発生するイベントです。
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。

基本的なプロパティ

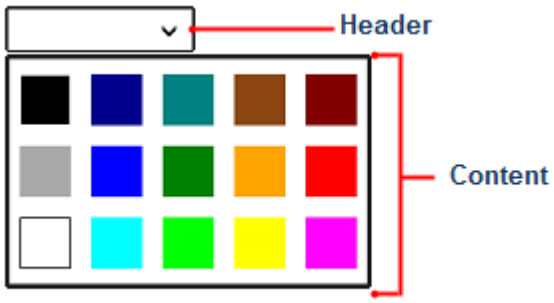
DropDown for UWP には、コントロールの機能を設定するためのいくつかのプロパティがあります。主要なプロパティを次に示します。外観を制御するプロパティの詳細については、「**DropDown for UWP** レイアウトおよび外観」を参照してください。

次のプロパティを使用して、**C1DropDown** コントロールをカスタマイズできます。

プロパティ	説明
AutoClose	ドロップダウン以外の場所をクリックすると自動的に閉じます。
AutoSizeMode	コンテンツのサイズが変化したときにポップアップのコンテンツのサイズを変更する方法を取得または設定します。
DropDownDirection	コントロールのドロップダウンボックスの展開方向を指定します。
DropDownHeight	ドロップダウンボックスの高さを取得または設定します。
DropDownWidth	ドロップダウンボックスの幅を取得または設定します。
ShowButton	ToggleButton が表示されるかどうかを取得または設定します。

C1DropDown の要素

C1DropDown コントロールは、ヘッダー領域とコンテンツ領域の2つのパーツで構成されます。ヘッダーはドロップダウンボックスが開いていないときに表示され、コンテンツはドロップダウン領域をクリックすると表示されます。次の図に、この2つのセクションを示します。



コンテンツは、まったく追加しないこともできますが、ヘッダー領域とコンテンツ領域の一方に追加することも両方に追加することもできます。XAML でコンテンツをヘッダー領域とコンテンツ領域に追加して、C1DropDown コントロールをカスタマイズできます。たとえば、次のマークアップでは、上の図のようなドロップダウンコントロールが作成されます。

```
<Xaml:C1DropDownButton Grid.Row="1" Background="White" x:Name="c1DropDown1" Padding="2" Width="150" HorizontalAlignment="Center" VerticalAlignment="Center" Xaml:C1NagScreen.Nag="True">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    <Xaml:C1DropDownButton.Header>
      <Border x:Name="dropDownBorder" Background="White" />
    </Xaml:C1DropDownButton.Header>
    <Xaml:C1DropDownButton.Content>
      <Xaml:C1TileListBox x:Name="colorListBox"
                          Height="180"
                          Orientation="Horizontal"
                          ItemTapped="colorListBox_ItemTapped"
                          SelectionMode="None"
                          BorderBrush="{StaticResource ComboBoxPopupBorderThemeBrush}"
                          BorderThickness="{StaticResource ComboBoxPopupBorderThemeThickness}"
                          Background="{StaticResource ComboBoxPopupBackgroundThemeBrush}">
        <Border Background="Black" BorderBrush="White" BorderThickness="1"/>
        <Border Background="DarkGray"/>
        <Border Background="White" BorderBrush="Black" BorderThickness="1"/>

        <Border Background="DarkBlue" />
        <Border Background="Blue" />
        <Border Background="Cyan" />

        <Border Background="Teal" />
        <Border Background="Green" />
        <Border Background="Lime" />

        <Border Background="SaddleBrown"/>
        <Border Background="Orange" />
        <Border Background="Yellow" />

        <Border Background="Maroon" />
        <Border Background="Red" />
        <Border Background="Magenta" />
      </Xaml:C1TileListBox>
    </Xaml:C1DropDownButton.Content>
  </Xaml:C1DropDownButton>
```

ヘッダーとコンテンツを定義するために `<Xaml:C1DropDown.Header>` タグと `<Xaml:C1DropDown.Content>` タグが使用されていることに注意してください。これらのタグの内側にコントロールとコンテンツを追加できます。

C1DropDown の操作

ユーザーは実行時に、ドロップダウンボックス内の項目または **C1DropDown** コントロール自体を操作できます。デフォルトでは、ユーザーはドロップダウンボックス内に配置されたコントロールを操作できます。たとえば、**C1DropDown** コントロール内にボタンまたはドロップダウンボックスを配置すると、ユーザーは実行時にそれをクリックできます。

DropDownDirection プロパティを使用すると、C1DropDown コントロールのドロップダウンの方向を制御できます。詳細については、「ドロップダウンボックスの方向」を参照してください。**IsDropDownOpen** プロパティを使用すると、**C1DropDown** ボックスを表示するときにドロップダウンボックスが自動的に開いた状態にするかどうかを選択できます。ユーザーがドロップダウンボックス以外の場所をクリックしたときにボックスを自動的に閉じるかどうかも設定できます。これは、**AutoClose** プロパティを使用して設定できます。

ドロップダウンボックスの方向

Copyright © GrapeCity, Inc. All rights reserved.

デフォルトでは、ユーザーが実行時に **C1DropDown** コントロールのドロップダウン矢印をクリックすると、コントロールの下にカラーピッカーが表示されます。コントロールの下に表示できない場合は、コントロールの上に表示されます。ただし、DropDownDirection プロパティを設定すると、カラーピッカーを表示する場所をカスタマイズできます。

DropDownDirection プロパティは、次のいずれかのオプションに設定できます。

イベント	説明
BelowOrAbove（デフォルト）	ドロップダウンボックスをヘッダーの下に表示するように試みます。表示できない場合は、その上に表示するように試みます。
AboveOrBelow	ドロップダウンボックスをヘッダーの上に表示するように試みます。表示できない場合は、その下に表示するように試みます。
ForceBelow	ドロップダウンボックスをヘッダーの下に強制的に表示します。
ForceAbove	ドロップダウンボックスをヘッダーの上に強制的に表示します。

詳細および例については、「ドロップダウンの方向の変更」を参照してください。

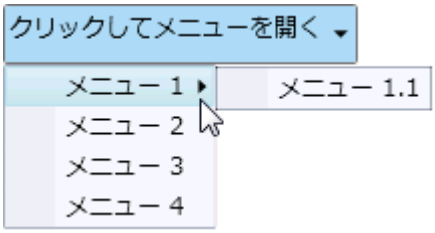
その他のコントロール

一般的な機能をすべて備えた **C1DropDown** コントロールのほかに、**DropDown for UWP** には、アプリケーションをさらに詳細にカスタマイズできるように C1DropDown コントロールのパーツと **C1DropDownButton** が含まれます。

C1DropDown は、従来のドロップダウンコントロールと同様に、いくつかの項目から選択を行うことができます。C1DropDownButton の機能はドロップダウンコントロールと同じですが、外観はボタンです。

C1DropDownButton の要素

C1DropDownButton コントロールは **C1DropDown** コントロールに似ており、ヘッダー領域とコンテンツ領域の2つの部分で構成されます。ヘッダーはドロップダウンボックスが開いていないときに表示され、コンテンツはドロップダウン領域をクリックすると表示されます。たとえば、下の図のコンテンツ領域には、構造化メニューが含まれます。



コンテンツは、まったく追加しないこともできますが、ヘッダー領域とコンテンツ領域の一方に追加することも両方に追加することもできます。XAML でコンテンツをヘッダー領域とコンテンツ領域に追加して、C1DropDown コントロールをカスタマイズできます。たとえば、次のマークアップでは、上の図のようなドロップダウンコントロールが作成されます。

```
<Xaml:C1DropDownButton x:Name="ddl" Header="メニューを開くには、ここをクリックします。"
HorizontalAlignment="Left" Xaml:C1NagScreen.Nag="True" Margin="451,301,0,391" Grid.Row="1">
        <Xaml:C1MenuList>
          <Xaml:C1MenuItem Header="メニュー 1">
            <Xaml:C1MenuItem Header="メニュー 1.1" />
          </Xaml:C1MenuItem>
          <Xaml:C1MenuItem Header="メニュー 2" />
          <Xaml:C1MenuItem Header="メニュー 3" />
          <Xaml:C1MenuItem Header="メニュー 4" />
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        </Xaml:C1MenuList>
    </Xaml:C1DropDownButton>
```

ヘッダーテキストは <c1:C1DropDownButton> タグで定義され、コンテンツは <c1:C1DropDownButton> </c1:C1DropDownButton> タグ内に置かれます。

レイアウトおよび外観

以下のトピックでは、**C1DropDown** 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。組み込みのレイアウトオプションを使用して、グリッドやキャンバスなどのコントロールをパネル内でレイアウトできます。テーマを使用して、ドロップダウンボックスの外観をカスタマイズできます。また、テンプレートを使用して、コントロールを書式設定およびレイアウトしたり、コントロールの操作をカスタマイズすることもできます。

色のプロパティ

次の色のプロパティを使用できます。

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	コントロールの前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。
HeaderBackground	コントロールのヘッダー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。
HeaderForeground	コントロールのヘッダーの前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。

配置プロパティ

次のプロパティを使用して、コントロールの配置をカスタマイズできます。

プロパティ	説明
HorizontalAlignment	パネルや項目コントロールなどの親要素内に置かれる要素に適用される水平配置の特性を取得または設定します。これは依存プロパティです。
VerticalAlignment	パネルや項目コントロールなどの親要素内に置かれる要素に適用される垂直配置の特性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

Copyright © GrapeCity, Inc. All rights reserved.

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
DropDownHeight	ドロップダウンボックスの高さを取得または設定します。
DropDownWidth	ドロップダウンボックスの幅を取得または設定します。
MaxDropDownHeight	ドロップダウンボックスの最大高さを取得または設定します。
MaxDropDownWidth	ドロップダウンボックスの最大幅を取得または設定します。
MinDropDownHeight	ドロップダウンボックスの最小高さを取得または設定します。
MinDropDownWidth	ドロップダウンボックスの最小幅を取得または設定します。

タスク別ヘルプ

次のタスクベースのヘルプトピックは、ユーザーの皆様が Visual Studio に精通しており、**C1DropDown** 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。**DropDown for UWP** 製品を初めて使用される場合は、まず「**DropDown for UWP** 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、**DropDown 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。タスクベースのヘルプの多くのトピックは、新しい 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 C1DropDown 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、「DropDown の作成」を参照してください。

DropDown の作成

C1DropDown コントロールは、設計時に、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 **C1DropDown** コントロールは、空で表示されます。コントロールに項目を追加する必要があります。例については、「**C1DropDown へのコンテンツの追加**」を参照してください。

XAML の場合

C1DropDown 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、**C1.Xaml.dll** アセンブリを選択して、[**OK**]をクリックします。
2. 最初の `<Page>` タグに `xmlns:Xaml="using:C1.Xaml"` を追加して、プロジェクトに XAML 名前空間を追加します。次のようになります。

マークアップ

```
<Page xmlns:Xaml="using:C1.Xaml"
    x:Class="DropDownTest.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:DropDownTest"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
```

3. グリッドの名前をLayoutRootに設定し、`<Xaml:C1DropDown>` タグをプロジェクトの `<Grid>` タグ内に追加して、C1DropDown コントロールを作成します。マークアップは次のようになります。

マークアップ

Copyright © GrapeCity, Inc. All rights reserved.

```
<Grid x:Name="LayoutRoot" Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
    <Xaml:C1DropDown Name="c1dropdown1" />
</Grid>
```

このマークアップは、「**c1dropdown1**」という名前の空の C1DropDown 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

C1DropDown 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、**C1.Xaml.dll** 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。
3. 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;
```

4. ページのコンストラクタに、C1DropDown 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。

C# コードの書き方

```
C#
C1DropDown c1dropdown1 = new C1DropDown();
c1dropdown1.Height = 30;
c1dropdown1.Width = 100;
LayoutRoot.Children.Add(c1dropdown1);
```

このコードは、「**c1dropdown1**」という名前の空の **C1DropDown** 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

C1DropDown コントロールを作成しました。上記の手順で作成した **C1DropDown** コントロールは、空で表示されます。実行時に操作できる項目をコントロールに追加できます。例については、「C1DropDown へのコンテンツの追加」を参照してください。

C1DropDown へのコンテンツの追加

C1DropDown コントロールには、任意の種類のコンテンツを追加できます。これには、テキスト、画像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、**C1DropDown** コントロールに **Button** 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、**Button** コントロールをドロップダウンに追加するには、`<Xaml:C1DropDown>` タグに `<Button Height="30" Name="button1" Width="100">`Hello World!`</Button>` を追加します。次のようになります。

```
マークアップ
<Xaml:C1DropDown HorizontalAlignment="Center" VerticalAlignment="Top" Width="100">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    <Xaml:C1DropDown.Content>
      <Button Height="30" Name="button1" Width="100">Hello World!</Button>
    </Xaml:C1DropDown.Content>
  </Xaml:C1DropDown>
```

コードの場合

たとえば、**Button** コントロールをドロップダウンボックスに追加するには、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。

C# コードの書き方

C#

```
public MainPage()
    {
        this.InitializeComponent();
        C1DropDown c1dropdown1 = new C1DropDown();
        c1dropdown1.Height = 30;
        c1dropdown1.Width = 100;
        LayoutRoot.Children.Add(c1dropdown1);
        Button c1button1 = new Button();
        c1button1.Content = "Hello World!";
        c1dropdown1.Content = c1button1;
    }
```

ここまでの成果

C1DropDown コントロールにボタン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、ドロップダウン矢印をクリックします。ドロップダウンボックスに Button コントロールが追加されていることを確認します。C1DropDown コントロールに複数の項目を追加する場合は、**C1DropDown** コントロールに Grid またはその他のパネルを追加してから、パネルに項目を追加します。

ドロップダウンの方向の変更

デフォルトでは、ユーザーが実行時に **C1DropDown** コントロールのドロップダウン矢印をクリックすると、コントロールの下にドロップダウンボックスが表示されます。コントロールの下に表示できない場合は、コントロールの上に表示されます。ただし、カラーピッカーを表示する場所をカスタマイズできます。ドロップダウン矢印の方向の詳細については、「ドロップダウンボックスの方向」を参照してください。

ドロップダウンの方向を変更するには、`<Xaml:C1DropDown>` タグに `DropDownDirection="ForceAbove"` を追加します。次のようになります。

XAML の場合

マークアップ

```
<Xaml:C1DropDown Height="30" Name="c1dropdown1" Width="100" DropDownDirection="ForceAbove"/>
```

コードの場合

ドロップダウンボックスの方向を変更するには、プロジェクトに次のコードを追加します。

C# コードの書き方

Copyright © GrapeCity, Inc. All rights reserved.

```
C#
c1dropdown1.DropDownDirection = DropDownDirection.ForceAbove;
```

ドロップダウン矢印の非表示

デフォルトでは、**C1DropDown** コントロールをアプリケーションに追加すると、ドロップダウン矢印（**ToggleButton**）が表示されます。ただし、ドロップダウン矢印は、次の手順に従って **ShowButton** プロパティを設定することにより、非表示にすることもできます。

XAML の場合

ドロップダウン矢印を非表示にするには、`<Xaml:C1DropDown>` タグに `ShowButton="False"`を追加します。次のようになります。

```
マークアップ
<XamlC1DropDown Height="30" Name="c1dropdown1" Width="100" ShowButton="False" / >
```

コードの場合

ドロップダウン矢印を非表示にするには、プロジェクトに次のコードを追加します。

C# コードの書き方

```
C#
c1dropdown1.ShowButton = false;
```

マウスオーバー時にドロップダウンを開く

デフォルトでは、**C1DropDown** コントロールのドロップダウンボックスは、ユーザーが実行時にドロップダウン矢印をクリックした場合にのみ開きます。このトピックでは、ユーザーが実行時にコントロールの上にマウスポインタを置くとドロップダウンボックスが開くように設定します。このトピックでは、アプリケーションにコンテンツを含む **C1DropDown** コントロールが既に追加されていることを前提としています。

次の手順を実行します。

1. C1DropDown コントロールを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、**Events** 稲妻ボタンをクリックして、コントロールに関連付けられているイベントを表示します。
3. IsMouseOverChanged 項目の横にあるボックスをダブルクリックし、コードビューに切り替えて **C1DropDown_IsMouseOverChanged** イベントハンドラを作成します。
4. 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;
   ```

5. **C1DropDown_IsMouseOver** 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。

   C# コードの書き方

Copyright © GrapeCity, Inc. All rights reserved.

C#

```
private void C1DropDown_IsMouseOverChanged(object sender, PropertyChangedEventArgs<bool> e)
      {
         if (c1dropdown1.IsMouseOver == true)
         {
            c1dropdown1.IsDropDownOpen = true;
         }
         else
         {
            c1dropdown1.IsDropDownOpen = false;
         }
      }
```

ここまでの成果

このトピックでは、**IsDropDownOpen** プロパティを使用して、ユーザーが実行時にコントロールの上にマウスポインタを置くとドロップダウンボックスが開くコードを追加しました。アプリケーションを実行し、コントロールの上にマウスポインタを置きます。ドロップダウンボックスが開くことを確認します。マウスポインタをコントロールから移動すると、ドロップダウンボックスが閉じることを確認します。

階層型 C1DropDown の作成

C1TreeView コントロールを格納した **C1DropDown** アプリケーションを作成して、簡単に階層型ドロップダウンを使用することができます。

階層型 **C1DropDown** コントロールを作成するには、次の手順に従います。

1. C1DropDown コントロールのマークアップを次のように編集します。

マークアップ

```
<Xaml:C1DropDownButton x:Name="soccerCountries" HorizontalAlignment="Center"
VerticalAlignment="Center" Padding="2" AutoClose="False" Width="150" DropDownWidth="200">
```

2. コントロールに **C1DropDown.Header** のコンテンツを追加します。ここでは、**C1DropDown** ヘッダーに **TextBlock** コントロールを追加します。

マークアップ

```
<Xaml:C1DropDownButton.Header>
            <TextBlock x:Name="selection" Text="« Pick one »" Padding="7 0 0 0" Foreground="Black"
TextAlignment="left"/>
         </Xaml:C1DropDownButton.Header>
```

3. `</Xaml:C1DropDownButton.Header>` タグの下にカーソルを置きます。Visual Studio ツールボックスの C1TreeView コントロールを見つけてダブルクリックし、このコントロールをアプリケーションに追加します。開始タグのマークアップを次のように編集します。

マークアップ

```
<Xaml:C1TreeView x:Name="treeSelection" Header="ワールドカップ優勝国"
KeyDown="C1TreeView_KeyDown" ItemClick="C1TreeView_ItemClicked" AllowDragDrop="False"
Padding="5"
                  BorderBrush="{StaticResource ComboBoxPopupBorderThemeBrush}"
                  BorderThickness="{StaticResource ComboBoxPopupBorderThemeThickness}"
                  Background="{StaticResource ComboBoxPopupBackgroundThemeBrush}">
```

このマークアップで、2つのイベント **KeyDown** と **ItemClick** を追加しました。

4. 次に、C1TreeView コントロールにコンテンツを追加します。

Copyright © GrapeCity, Inc. All rights reserved.

マークアップ

```
<Xaml:C1TreeViewItem Header="南アメリカ">
  <Xaml:C1TreeViewItem Header="アルゼンチン" />
  <Xaml:C1TreeViewItem Header="ブラジル" />
  <Xaml:C1TreeViewItem Header="ウルグアイ" />
</Xaml:C1TreeViewItem>
<Xaml:C1TreeViewItem Header="ヨーロッパ">
  <Xaml:C1TreeViewItem Header="イングランド" />
  <Xaml:C1TreeViewItem Header="フランス" />
  <Xaml:C1TreeViewItem Header="ドイツ" />
  <Xaml:C1TreeViewItem Header="イタリア" />
  <Xaml:C1TreeViewItem Header="スペイン" />
</Xaml:C1TreeViewItem>
```

5. アプリケーションのマークアップを作成できたので、ページを右クリックし、リストから[**コードの表示**]を選択します。コードビューが開きます。
6. 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;
```

7. **InitializeComponent** メソッドの閉じかっこの後に次の **KeyDown** イベントを追加します。

C# コードの書き方

C#

```
private void C1TreeView_KeyDown(object sender, KeyRoutedEventArgs e)
      {
         if (e.Key == VirtualKey.Enter)
         {
            UpdateSelection();
            e.Handled = true;
         }
      }
```

8. 次に、**UpdateSelection()** メソッドを追加します。

C# コードの書き方

C#

```
private void UpdateSelection()
      {
         if (treeSelection.SelectedItem != null)
         {
            selection.Text = treeSelection.SelectedItem.Header.ToString();
         }
         else
         {
            selection.Text = "« 国を選択 »";
         }
         soccerCountries.IsDropDownOpen = false;
      }
```


Copyright © GrapeCity, Inc. All rights reserved.

9. 次に、**C1TreeView_ItemClicked** イベントを作成します。

C# コードの書き方

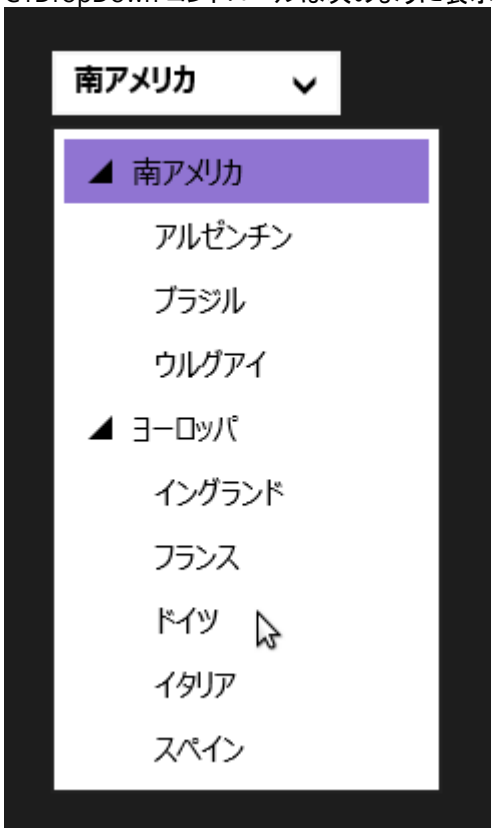
```
C#
private void C1TreeView_ItemClicked(object sender, SourcedEventArgs e)
    {
        UpdateSelection();
    }
```

10. 最後に、**MouseLeftButtonDown** イベントを追加します。

C# コードの書き方

```
C#
private void ContentControl_MouseLeftButtonDown(object sender, EventArgs e)
    {
        soccerCountries.IsDropDownOpen = true;
    }
```

11. [F5]キーを押すか、デバッグを開始して、アプリケーションを実行します。ドロップダウンボタンをクリックすると、C1DropDown コントロールは次のように表示されます。大陸を選択すると、その大陸にある国が表示されます。



ここまでの成果

このトピックでは、XAML マークアップとコードを使用して階層型 C1DropDown コントロールを作成しました。

Layout Panels for UWP

Layout Panels for UWP を使用して UWP アプリケーションのコンテンツのフローと配置を制御します。C1WrapPanel を使用すると、コンテンツを垂直または水平方向に折り返すことができます。C1DockPanel を使用すると、パネルの端にコンテンツを

Copyright © GrapeCity, Inc. All rights reserved.

ドッキングできます。C1UniformGrid を使用すると、コンテンツをグリッドに表示できます。

主な特長

Layout Panels for UWP には、次の主な特長があります。

- **フローレイアウトの作成**
  C1WrapPanel コントロールを使用すると、コンテンツを垂直または水平方向に折り返すフロータイプのレイアウトを作成できます。これは、ユーザーがアプリケーションを回転させて縦方向にしたときに項目フローを処理するためにたいへん便利です。
- **ドッキングレイアウトの作成**
  C1DockPanel コントロールを使用すると、コンテンツを画面の上下左右の端にドッキングできます。子要素は、XAML で宣言された順序でドッキングパネルに配置されます。
- **ユニフォームグリッドレイアウトの作成**
  C1UniformGrid コントロールを使用すると、列と行に子要素を並べて表示できます。**ColumnSpan** プロパティを設定して、複数の列を結合できます。また、**RowSpan** プロパティを設定して、複数の行を結合できます。これは、組み込みの Microsoft グリッドに似ています。また、C1UniformGrid を使用すると、ある列全体または行全体を表示または非表示にすることができます。

クイックスタート

Layout Panels for UWP のコントロールごとにクイックスタートが作成されています。各クイックスタートでは、最初に UWP アプリケーションを作成し、次にコントロールのスタイルを追加します。**WrapPanel** では、複数の **HyperlinkButtons** を折り返します。**DockPanel** では、複数の要素を持つパネルを作成し、C1DockPanel の4つのドッキングオプションを示します。**UniformGrid** では、3つの列を持つグリッドが作成され、最初の行に2つの空のセルが作成されます。

クイックスタートを開始するには、次のいずれかを選択します。

- WrapPanel クイックスタート
- DockPanel クイックスタート
- UniformGrid クイックスタート

WrapPanel クイックスタート

このクイックスタートガイドは、**WrapPanel 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、Visual Studio で新しいプロジェクトを作成し、スタイルが設定された折り返し可能な HyperlinkButtons を追加して、ボタンの並ぶ方向を変更します。

手順1：アプリケーションの作成

この手順では、最初に Visual Studio で **WrapPanel for UWP** を使用する UWP スタイルのアプリケーションを作成します。

プロジェクトを設定するには、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、言語の下で[**Windows ストア**]を選択し、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. **MainPage.xaml** をまだ開いていない場合は開きます。

次のステップでは、複数の **HyperlinkButtons** ボタンを追加し、スタイルを設定し、折り返します。

Copyright © GrapeCity, Inc. All rights reserved.

手順2：アプリケーションへの C1WrapPanel の追加

シンプルな **HyperlinkButtons** ボタンを使用して、コンテンツを垂直または水平方向に折り返す方法について説明します。これは Web アプリケーションで頻繁に使用される TagCloud ビューを作成する一般的なシナリオです。

次の手順に従います。

1. 最初に、プロジェクトから **Grid** タグを取り除きます。
2. C1WrapPanelコントロールをページにドラッグ &ドロップします。これにより、パネルと参照がページに追加されます。
3. `<Xaml:C1WrapPanel>` タグを編集し、**HyperlinkButtons** を追加します。マークアップは次のようになります。

マークアップ

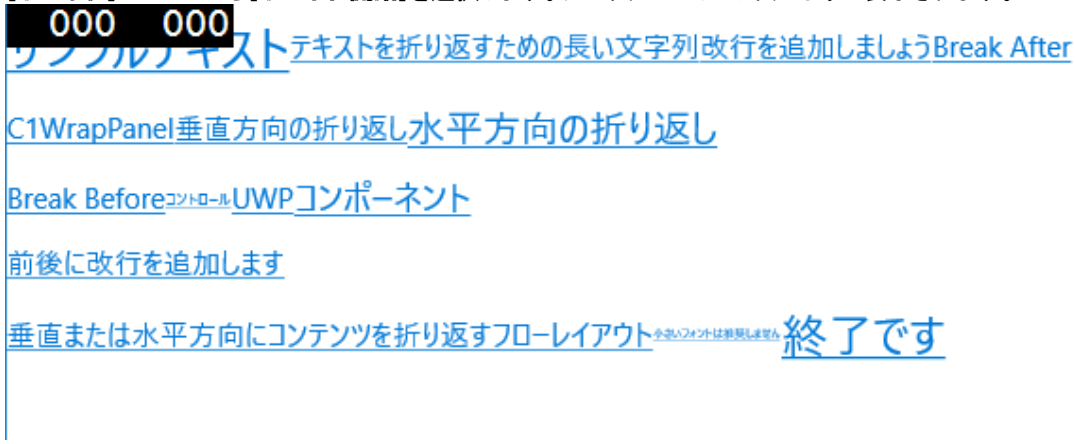
```
<Xaml:C1WrapPanel>
        <HyperlinkButton Content="サンプルテキスト" FontSize="25" />
        <HyperlinkButton Content="テキストを折り返すための長い文字列" />
        <HyperlinkButton Content="改行を追加しましょう" />
        <HyperlinkButton Xaml:C1WrapPanel.BreakLine="After" Content="後ろに改行を追加します" />
        <HyperlinkButton Content="C1WrapPanel" />
        <HyperlinkButton Content="垂直方向の折り返し" />
        <HyperlinkButton Content="水平方向の折り返し" FontSize="20"  />
        <HyperlinkButton Xaml:C1WrapPanel.BreakLine="Before" Content="Break Before" />
        <HyperlinkButton Content="コントロール" FontSize="8" />
        <HyperlinkButton Content="UWP" />
        <HyperlinkButton Content="コンポーネント" FontSize="18" />
        <HyperlinkButton Xaml:C1WrapPanel.BreakLine="AfterAndBefore" Content="前後に改行を追加します"
/>
        <HyperlinkButton Content="垂直または水平方向にコンテンツを折り返すフローレイアウト" />
        <HyperlinkButton Content="小さいフォントは推奨しません" FontSize="6" />
        <HyperlinkButton Content="終了です" FontSize="24" />
    </Xaml:C1WrapPanel>
```

次の手順では、このアプリケーションを実行します。

手順3：アプリケーションの実行

これで、アプリケーションを実行する準備ができました。次の手順に従います。

1. [**デバッグ**]メニューから[**デバッグ開始**]を選択します。アプリケーションは次のように表示されます。



2. [**デバッグの停止**]ボタンをクリックしてアプリケーションを終了します。
3. **MainPage.xaml** に戻ります。`<Xaml:C1WrapPanel>` タグで、**Orientation** プロパティを **Vertical** に設定します。XAML は次のようになります。

Copyright © GrapeCity, Inc. All rights reserved.

マークアップ

```
<Xaml:C1WrapPanel Orientation="Vertical">
```

4. [**デバッグ**]メニューで再度[**デバッグ開始**]をクリックします。アプリケーションは次のようになります。

ボタンが垂直方向に積み重なっています。

おめでとうございます。これで、**WrapPanel for UWP** クイックスタートは終了です。

DockPanel クイックスタート

このクイックスタートガイドは、**DockPanel 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、Visual Studio で新しいプロジェクトを作成し、C1DockPanel の上下左右にドッキングする要素を追加します。

手順1：アプリケーションの作成

この手順では、最初に Visual Studio で **DockPanel for UWP** を使用する UWP スタイルのアプリケーションを作成します。

プロジェクトを設定するには、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、言語の下で[**Windows ストア**]を選択し、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. **MainPage.xaml** をまだ開いていない場合は開きます。

次の手順では、**C1DockPanels** を追加してカスタマイズします。

手順2：アプリケーションへの C1DockPanel の追加

この手順では、複数の **C1DockPanels** を追加してスタイルを設定します。

次の手順に従います。

1. 最初に、プロジェクトから **Grid** タグを取り除きます。
2. C1DockPanelコントロールをページにドラッグ &ドロップします。これにより、パネルと参照がページに追加されます。
3. `<Xaml:C1DockPanel>` タグを編集して、画面の上下左右にドッキングされる境界線を追加します。マークアップは次のようになります。

マークアップ

```
<Xaml:C1DockPanel Background="White" Width="400" Height="250">
  <Border Xaml:C1DockPanel.Dock="Top" Height="50" Background="Red">
    <TextBlock Text="上" />
  </Border>
  <Border Xaml:C1DockPanel.Dock="Bottom" Height="50" Background="Blue">
    <TextBlock Text="下" />
  </Border>
```

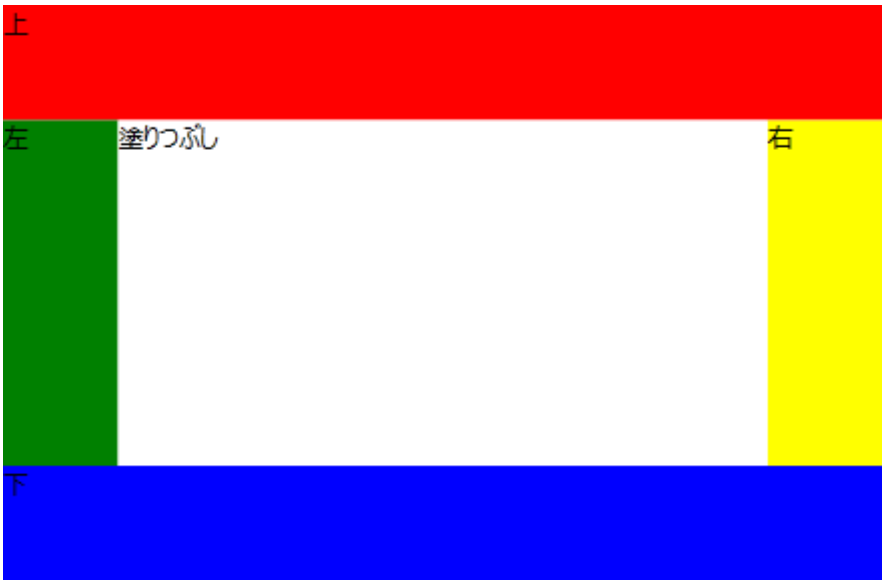

Copyright © GrapeCity, Inc. All rights reserved.

```
  <Border Xaml:C1DockPanel.Dock="Right" Width="50" Background="Yellow">
    <TextBlock Text="右" />
  </Border>
  <Border Xaml:C1DockPanel.Dock="Left" Width="50" Background="Green">
    <TextBlock Text="左" />
  </Border>
</Xaml:C1DockPanel>
```

次の手順では、このアプリケーションを実行します。

手順3：アプリケーションの実行

これで、アプリケーションを実行する準備ができました。[**デバッグ**]メニューから[**デバッグ開始**]を選択します。アプリケーションは次のようになります。**C1DockPanel** コントロールの上下左右に4つの境界線がドッキングされています。



おめでとうございます。これで、**DockPanel for UWP** クイックスタートは終了です。

UniformGrid クイックスタート

このクイックスタートガイドは、**UniformGrid 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、Visual Studio で新しいプロジェクトを作成し、**C1UniformGrid** をアプリケーションに追加し、**Columns**、**FirstColumn**、Width の各プロパティを設定してから、アプリケーションを実行します。

手順1：UWP アプリケーションの作成

この手順では、最初に Visual Studio で **UniformGrid for UWP** を使用する UWP スタイルのアプリケーションを作成します。

プロジェクトを設定するには、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、言語の下で[**Windows ストア**]を選択し、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. **MainPage.xaml** をまだ開いていない場合は開きます。

次のステップでは、複数の境界線ブロックを追加し、スタイルを設定し、折り返します。

Copyright © GrapeCity, Inc. All rights reserved.

手順2：アプリケーションへの C1UniformGrid の追加

この手順では、C1UniformGrid を追加します。

次の手順に従います。

1. 最初に、プロジェクトから **Grid** タグを取り除きます。
2. C1UniformGrid コントロールをページにドラッグ &ドロップします。これで、パネルと参照がページに追加されます。
3. `<Xaml:C1UniformGrid>` タグを編集して、グリッドのサイズを指定し、子要素（この場合は番号の付いたセル）をグリッドに追加します。マークアップは次のようになります。

マークアップ

```
<Xaml:C1UniformGrid Width="300" Height="300">
        <Border Background="#FF005B84" >
          <TextBlock Text="0" />
        </Border>
        <Border Background="#FF008B9C" >
          <TextBlock Text="1" />
        </Border>
        <Border Background="#FF00ADD6" >
          <TextBlock Text="2" />
        </Border>
        <Border Background="#FF497331" >
          <TextBlock Text="3" />
        </Border>
        <Border Background="#FF0094D6" >
          <TextBlock Text="4" />
        </Border>
        <Border Background="#FF9DCFC3" >
          <TextBlock Text="5" />
        </Border>
        <Border Background="#FFA5DDFE" >
          <TextBlock Text="6" />
        </Border>
        <Border Background="#FFE0EEEF" >
          <TextBlock Text="7" />
        </Border>
        <Border Background="CornflowerBlue" >
          <TextBlock Text="8" />
        </Border>
        <Border Background="LightYellow" >
          <TextBlock Text="9" />
        </Border>
  </Xaml:C1UniformGrid>
```

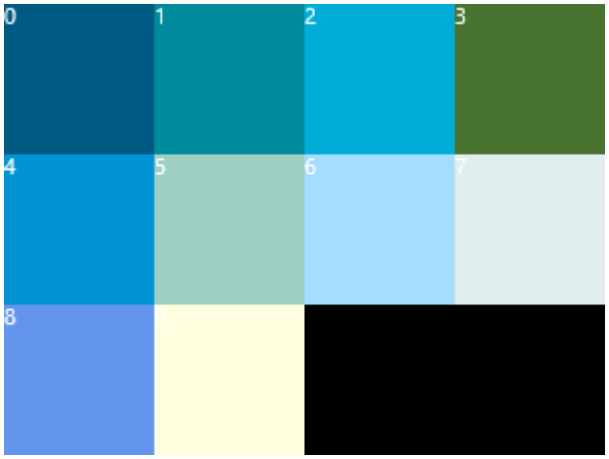
4. この手順では、C1UniformGrid パネルにコンテンツを追加しました。次の手順では、アプリケーションを実行し、グリッドに追加したプロパティのいくつかを編集します。

手順3：アプリケーションの実行

これで、アプリケーションを実行する準備ができました。次の手順に従います。

1. [**デバッグ**]メニューから[**デバッグ開始**]を選択します。アプリケーションは次のように表示されます。

Copyright © GrapeCity, Inc. All rights reserved.

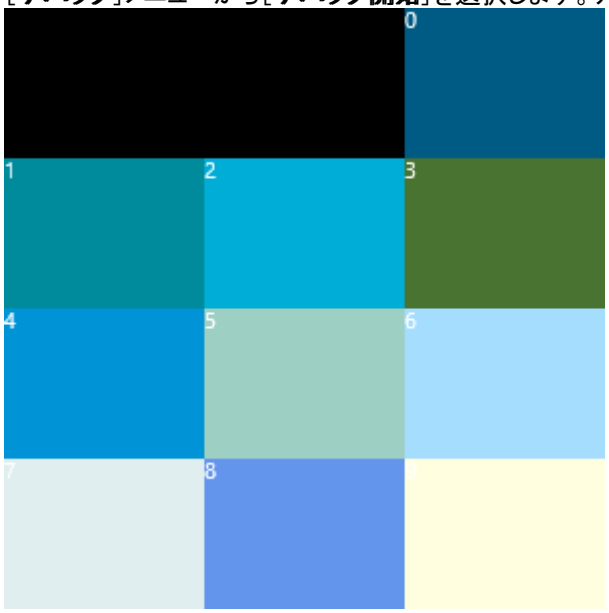


2. [**デバッグの停止**]ボタンをクリックしてアプリケーションを終了します。
3. **MainPage.xaml** に戻り、カーソルを `<Xaml:C1UniformGrid>` タグに置きます。
4. 次の XAML マークアップを使用して、**Columns** および **FirstColumn** プロパティを設定します。

```
<Xaml:C1UniformGrid Width="300" Height="300" Columns="3" FirstColumn="2" >
```

Columns は、グリッド内の列の数を設定し、Width プロパティは幅をピクセル単位で設定し、FirstColumn プロパティは最初の行に表示する空のセルの数を設定します。

5. [**デバッグ**]メニューから[**デバッグ開始**]を選択します。アプリケーションは次のように表示されます。



FirstColumn プロパティの指定に基づいて、最初の行に空のセルが2つできることを確認します。

おめでとうございます。これで、**UniformGrid for UWP** クイックスタートは終了です。

タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio .NET でのプログラミングに精通しており、Layout Panels 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。**Layout Panels 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、**Layout Panels for UWP** 製品を使用して特定のタスクを行うための方法を提供します。

また、タスクベースの各ヘルプトピックは、新しい UWP プロジェクトが既に作成されていることを前提としています。

Copyright © GrapeCity, Inc. All rights reserved.

C1WrapPanel で項目を折り返す

C1WrapPanel.BreakLine 添付プロパティを使用すると、項目を折り返すことができます。この例では、**HyperlinkButtons** を使用します。次の手順に従います。

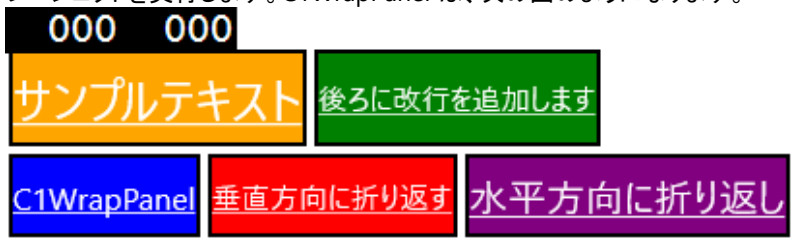
1. プロジェクトで、ツールボックスから C1WrapPanel コントロールをドラッグし、.xaml ページの `</Grid>` 終了タグの前に置きます。
2. `<Xaml:C1WrapPanel>` タグの間にカーソルを置き、[Enter]キーを押します。
3. 次の XAML を追加して **HyperlinkButtons** を折り返します。

マークアップ

```
<Border Margin="2" BorderBrush="Black" BorderThickness="2" Background="Orange">
   <HyperlinkButton Foreground="White"  Content="サンプルテキスト" FontSize="25" />
</Border>
<Border Margin="2" BorderBrush="Black" BorderThickness="2" Background="Green"
Xaml:C1WrapPanel.BreakLine="After">
   <HyperlinkButton Foreground="White" Content="後ろに改行を追加します" />
</Border>
<Border Margin="2" BorderBrush="Black" BorderThickness="2" Background="Blue">
   <HyperlinkButton Foreground="White" Content="C1WrapPanel" FontSize="16"/>
</Border>
<Border Margin="2" BorderBrush="Black" BorderThickness="2" Background="Red">
   <HyperlinkButton Foreground="White" Content="UWP" />
</Border>
<Border Margin="2" BorderBrush="Black" BorderThickness="2"  Background="Purple">
   <HyperlinkButton Foreground="White" Content="水平方向に折り返し" FontSize="20" />
</Border>
```

2番目の **HyperlinkButton** では、**C1WrapPanel.BreakLine** プロパティは "**After**" に設定されています。これにより、ボタンの後に改行が追加されます。

4. プロジェクトを実行します。C1WrapPanel は、次の図のようになります。



2番目の **HyperlinkButton** の後に改行があります。

C1WrapPanel で項目を垂直方向に折り返す

デフォルトでは、項目は水平方向に折り返されます。ただし、垂直方向に折り返すこともできます。垂直方向の折り返しを指定するには、**Orientation** プロパティを設定します。この例では、**HyperlinkButtons** を使用します。次の手順に従います。

1. プロジェクトで、ツールボックスから C1WrapPanel コントロールをドラッグし、.xaml ページの `</Grid>` 終了タグの前に置きます。

Copyright © GrapeCity, Inc. All rights reserved.

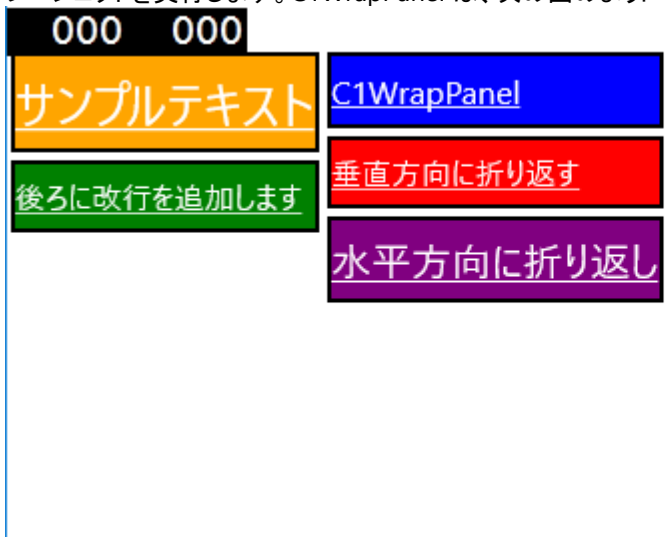
2. <Xaml:C1WrapPanel> タグで、Orientation プロパティを **Vertical** に設定します。XAML は次のようになります。

マークアップ
<Xaml:C1WrapPanel Orientation="Vertical">

3. <Xaml:C1WrapPanel> タグの間にカーソルを置き、[Enter]キーを押します。
4. 次の XAML を追加して HyperlinkButtons を折り返します。

```
マークアップ
<Border Margin="2" BorderBrush="Black" BorderThickness="2" Background="Orange">
   <HyperlinkButton Foreground="White" Content="サンプルテキスト" FontSize="25" />
</Border>
<Border Margin="2" BorderBrush="Black" BorderThickness="2" Background="Green"
Xaml:C1WrapPanel.BreakLine="After">
   <HyperlinkButton Foreground="White" Content="後ろに改行を追加します" />
</Border>
<Border Margin="2" BorderBrush="Black" BorderThickness="2" Background="Blue">
   <HyperlinkButton Foreground="White" Content="C1WrapPanel" FontSize="16"/>
</Border>
<Border Margin="2" BorderBrush="Black" BorderThickness="2" Background="Red">
   <HyperlinkButton Foreground="White" Content="垂直方向に折り返し" />
</Border>
<Border Margin="2" BorderBrush="Black" BorderThickness="2"  Background="Purple">
   <HyperlinkButton Foreground="White" Content="UWP" FontSize="20"/>
</Border>
```

5. プロジェクトを実行します。C1WrapPanel は、次の図のようになります。



ListBox for UWP

連結データをリスト表示する場合は、**ListBox for UWP** に含まれる高パフォーマンスな2つのコントロールを利用してください。**C1ListBox** コントロールや **C1TileListBox** コントロールを使用して、リストをタイル状にレイアウトして表示したり、光学ズーム付きで表示することができます。これらのコントロールは UI の仮想化をサポートしています。そのため、動作は極めて高速であり、数千個の項目を表示してもパフォーマンスはほとんど低下しません。

主な特長

ListBox for UWP には、次の主な特長があります。

Copyright © GrapeCity, Inc. All rights reserved.

- **水平方向または垂直方向**
  ListBox コントロールは、水平方向と垂直方向の両方をサポートしています。このため、より多くのレイアウトシナリオに対応することができます。
- **項目のタイル表示**
  C1TileListBox では、項目を複数の行と列に並べて、タイル状に表示することができます。各項目のサイズとテンプレートを指定し、任意の方向を選択します。
- **光学ズーム**
  C1ListBox コントロールは、光学ズーム機能をサポートしています。ユーザーは、ピンチジェスチャを使用して、項目のサイズ変更操作を直観的に行うことができます。ズーム動作は流れるようにスムースです。アプリケーションのパフォーマンスが犠牲になることはありません。
- **UI の仮想化**
  C1ListBox コントロールは UI の仮想化をサポートしています。そのため、動作は極めて高速であり、数千個の項目を表示してもパフォーマンスはほとんど低下しません。各レイアウトパスにレンダリングする項目の数を決定するには、**ViewportGap** プロパティと **ViewportPreviewGap** プロパティを設定します。これらのプロパティをシナリオに応じて調整できます。
- **プレビュー状態**
  パフォーマンスを可能な限り高めるために、C1ListBox コントロールは、ビューポートの外の項目をプレビュー状態でレンダリングすることができます。標準の **ItemTemplate** と同様に、**Preview** テンプレートは、ズームアウト中または高速スクロール中などのプレビュー状態にある項目の外観を定義します。その後、項目のスクロールやズームが終了すると、完全な項目テンプレートに切り替えられます。

C1ListBox クイックスタート

このクイックスタートガイドは、**C1ListBox** コントロール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ガイドでは、最初に Visual Studio で新しいプロジェクトを作成し、アプリケーションに C1ListBox 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：C1ListBox コントロールを含むアプリケーションの作成

この手順では、Visual Studio で、**ListBox for UWP** を使用して UWP アプリケーションを作成します。

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、言語の下で[**Windows ストア**]を選択し、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. ソリューションエクスプローラでプロジェクト名を右クリックし、[**参照の追加**]を選択します。
4. [**参照マネージャ**]ダイアログボックスで[**ComponentOne for UWP**]を選択します。[**OK**]をクリックしてダイアログボックスを閉じ、参照を追加します。
5. **MainPage.xaml** をまだ開いていない場合は開き、`<Page>` タグ内に次のマークアップを追加します。

```
xmlns:c1="using:C1.Xaml"
xmlns:c1tile="using:C1.Xaml.Tile"
```

   これにより、C1.Xaml および C1.Xaml.Tile アセンブリへの参照がプロジェクトに追加されます。
6. `<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。
7. `<Grid>` タグと `</Grid>` タグの間に次の `<StackPanel>` マークアップを追加して、**TextBlock** と **ProgressBar** を含む **StackPanel** を追加します。

```
<StackPanel x:Name="loading" VerticalAlignment="Center">
    <TextBlock Text="Retrieving data from Flickr..." TextAlignment="Center"/>
    <ProgressBar IsIndeterminate="True" Width="200" Height="4"/>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
</StackPanel>
```

この **TextBlock** と **ProgressBar** は、C1ListBox が読み込み中であることを示します。

8. ツールボックスに移動し、**C1ListBox** 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。
9. `<Xaml:C1ListBox>` タグを編集して、コントロールをカスタマイズします。

```
<c1:C1ListBox x:Name="listBox" ItemsSource="{Binding}" Background="Transparent" Visibility="Collapsed"
ItemWidth="800" ItemHeight="600" RefreshWhileScrolling="False"></c1:C1ListBox>
```

これは、コントロールに名前を付け、コントロールの連結、背景、表示/非表示、サイズ、および更新機能をカスタマイズします。

10. `<Xaml:C1ListBox>` タグと `</Xaml:C1ListBox>` タグの間に次のマークアップを追加します。

```
<c1:C1ListBox.PreviewItemTemplate>
  <DataTemplate>
    <Grid Background="Gray">
      <Image Source="{Binding Thumbnail}" Stretch="UniformToFill" />
    </Grid>
  </DataTemplate>
</c1:C1ListBox.PreviewItemTemplate>
<c1:C1ListBox.ItemTemplate>
  <DataTemplate>
    <Grid>
      <Image Source="{Binding Content}" Stretch="UniformToFill" />
      <TextBlock Text="{Binding Title}" Foreground="White" Margin="4 0 0 4"
VerticalAlignment="Bottom" />
    </Grid>
  </DataTemplate>
</c1:C1ListBox.ItemTemplate>
```

11. このマークアップは、**C1ListBox** コントロールのコンテンツのデータテンプレートを追加します。このコントロールの連結はコードで行います。

ここまでの成果

これで、**C1ListBox** コントロールを含む UWP スタイルのアプリケーションを作成できました。次の「手順2：ListBox へのデータの追加」では、**C1ListBox** にデータを追加します。

手順2：ListBox へのデータの追加

前の手順では、**C1ListBox** コントロールをアプリケーションに追加しました。この手順では、フォトストリームから画像を表示するコードを追加します。

プログラムでコントロールにデータを追加するには、次の手順に従います。

1. [**ページ**]を選択して[**プロパティ**]ウィンドウに移動し、稲妻の[**イベント**]ボタンをクリックしてイベントを表示します。次に、下にスクロールして、**Loaded** イベントの横にある領域をダブルクリックします。
   これで、コードエディタが開き、**Page_Loaded** イベントが追加されます。
2. 次の **imports** 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
```


Copyright © GrapeCity, Inc. All rights reserved.

```
Imports C1.Xaml
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Net
Imports System.Xml.Linq
Imports Windows.UI.Popups
Imports Windows.UI.Xaml
Imports Windows.UI.Xaml.Controls
```

C# コードの書き方

C#

```
using C1.Xaml;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Xml.Linq;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
```

3. **Page_Loaded** イベントハンドラ内に次のコードを追加します。

Visual Basic コードの書き方

Visual Basic

```
LoadPhotos()
```

C# コードの書き方

C#

```
LoadPhotos();
```

4. **MainPage** クラス内の **Page_Loaded** イベントの下に次のコードを追加します。

Visual Basic コードの書き方

Visual Basic

```
Private Async Function LoadPhotos() As Task
    Dim flickrUrl = "http://api.flickr.com/services/feeds/photos_public.gne"
    Dim AtomNS = "http://www.w3.org/2005/Atom"

    Dim photos = New List(Of Photo)()
    Dim client = WebRequest.CreateHttp(New Uri(flickrUrl))
    Dim response = Await client.GetResponseAsync()

    Try
        '#Region "** parse data"
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        Dim doc = XDocument.Load(response.GetResponseStream())
        For Each entry As XElement In doc.Descendants(XName.[Get]("entry", AtomNS))
            Dim title = entry.Element(XName.[Get]("title", AtomNS)).Value


            Dim enclosure = entry.Elements(XName.[Get]("link", AtomNS)).Where(Function(elem)
elem.Attribute("rel").Value = "enclosure").FirstOrDefault()
            Dim contentUri = enclosure.Attribute("href").Value
            photos.Add(New Photo() With { _
                .Title = title, _
                .Content = contentUri, _
                .Thumbnail = contentUri.Replace("_b", "_m") _
            })
        Next
        '#End Region


        listBox.ItemsSource = photos
        loading.Visibility = Visibility.Collapsed
        listBox.Zoom = C1ZoomUnit.Fill
        listBox.Visibility = Visibility.Visible
    Catch
        Dim dialog = New MessageDialog("There was an error when attempting to download data from
Flickr.")
        async dialog.ShowAsync()
    End Try
  End Function
```

C# コードの書き方

C#

```
private async void LoadPhotos()
    {
        var flickrUrl = "http://api.flickr.com/services/feeds/photos_public.gne";
        var AtomNS = "http://www.w3.org/2005/Atom";


        var photos = new List<Photo>();
        var client = WebRequest.CreateHttp(new Uri(flickrUrl));
        var response = await client.GetResponseAsync();


        try
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        {
            #region ** parse data
            var doc = XDocument.Load(response.GetResponseStream());
            foreach (var entry in doc.Descendants(XName.Get("entry", AtomNS)))
            {
                var title = entry.Element(XName.Get("title", AtomNS)).Value;

                var enclosure = entry.Elements(XName.Get("link", AtomNS)).Where(elem =>
elem.Attribute("rel").Value == "enclosure").FirstOrDefault();
                var contentUri = enclosure.Attribute("href").Value;
                photos.Add(new Photo() { Title = title, Content = contentUri, Thumbnail =
contentUri.Replace("_b","_m") });
            }
            #endregion

            listBox.ItemsSource = photos;
            loading.Visibility = Visibility.Collapsed;
            listBox.Zoom = C1ZoomUnit.Fill;
            listBox.Visibility = Visibility.Visible;
        }
        catch
        {
            var dialog = new MessageDialog("There was an error when attempting to download data from
Flickr.");
            async dialog.ShowAsync();
        }
    }
```

5. 上のコードは、Flickr のパブリックフォトストリームから画像を取得し、それらの画像のリストに **C1ListBox** を連結します。
6. **MainPage** クラスの直後に次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
Public Class Photo
    Public Property Title() As String
        Get
            Return m_Title
        End Get
        Set(value As String)
            m_Title = Value
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        End Set
    End Property
    Private m_Title As String
    Public Property Thumbnail() As String
        Get
            Return m_Thumbnail
        End Get
        Set(value As String)
            m_Thumbnail = Value
        End Set
    End Property
    Private m_Thumbnail As String
    Public Property Content() As String
        Get
            Return m_Content
        End Get
        Set(value As String)
            m_Content = Value
        End Set
    End Property
    Private m_Content As String
End Class
```

C# コードの書き方

C#

```
public class Photo
  {
    public string Title { get; set; }
    public string Thumbnail { get; set; }
    public string Content { get; set; }
  }
```

ここまでの成果

これで、**C1TileListBox** にデータを追加できました。次の「手順3：ListBox アプリケーションの実行」では、**ListBox for UWP** の機能について説明します。

手順3：ListBox アプリケーションの実行

これまでに、UWP スタイルアプリケーションを作成し、外観と動作をカスタマイズしたので、次にアプリケーションを実行しま

Copyright © GrapeCity, Inc. All rights reserved.

す。アプリケーションを実行し、**ListBox 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. [**デバッグ**]メニューから[**デバッグ開始**]を選択し、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。アプリケーションが表示され、画像が表示されます。
2. コントロールの右側にあるスクロールバーを使用して、イメージストリームをスクロールします。
3. タッチ機能がある場合は、ピンチして画像をズームしてみてください。

ここまでの成果

おめでとうございます。これで **ListBox for UWP** クイックスタートは完了です。C1ListBox コントロールを使用するアプリケーションを作成し、アプリケーションの実行時機能をいくつか確認することができました。

C1TileListBox クイックスタート

このクイックスタートガイドは、**C1TileListBox**コントロール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ガイドでは、最初に Visual Studio で新しいプロジェクトを作成し、アプリケーションに **C1TileListBox** 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：C1TileListBox コントロールを含むアプリケーションの作成

この手順では、Visual Studio で、**TileListBox for UWP** を使用して UWP アプリケーションを作成します。

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、言語の下で[**Windows ストア**]を選択し、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. **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。
4. `<Grid>` タグと `</Grid>` タグの間に次の `<StackPanel>` マークアップを追加して、**TextBlock** と **ProgressBar** を含む **StackPanel** を追加します。

```
<StackPanel x:Name="loading" VerticalAlignment="Center">
   <TextBlock Text="YouTubeからデータを取得しています..." TextAlignment="Center"/>
   <ProgressBar IsIndeterminate="True" Width="200" Height="4"/>
 </StackPanel>
```

   この **TextBlock** と **ProgressBar** は、**C1TileListBox** が読み込み中であることを示します。
5. ツールボックスに移動し、C1TileListBox 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。
6. `<Xaml:C1TileListBox>` タグを編集して、コントロールをカスタマイズします。

```
<Xaml:C1TileListBox x:Name="tileListBox" ItemsSource="{Binding}" Background="Transparent"
Visibility="Collapsed" ItemWidth="800" ItemHeight="600" RefreshWhileScrolling="False">
</Xaml:C1TileListBox>
```

   これは、コントロールに名前を付け、コントロールの連結、背景、表示/非表示、サイズ、および更新機能をカスタマイズします。
7. `<Xaml:C1TileListBox>` タグと `</Xaml:C1TileListBox>` タグの間に次のマークアップを追加します。

```
<Xaml:C1TileListBox.PreviewItemTemplate>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        <DataTemplate>
            <Grid Background="Gray"/>
        </DataTemplate>
            </Xaml:C1TileListBox.PreviewItemTemplate>
            <Xaml:C1TileListBox.ItemTemplate>
        <DataTemplate>
        <Grid Background="LightBlue">
            <Image Source="{Binding Thumbnail}" Stretch="UniformToFill"/>
            <TextBlock Text="{Binding Title}" Foreground="White" Margin="4 0 0 4"
VerticalAlignment="Bottom"/>
            </Grid>
        </DataTemplate>
    </Xaml:C1TileListBox.ItemTemplate>
```

8. このマークアップは、**C1TileListBox** コントロールのコンテンツのデータテンプレートを追加します。このコントロールの連結はコードで行います。

ここまでの成果

これで、C1TileListBox コントロールを含む UWP スタイルのアプリケーションを作成できました。次の「手順2：TileListBox へのデータの追加」では、**C1TileListBox** にデータを追加します。

手順2：TileListBox へのデータの追加

前の手順では、**C1TileListBox**コントロールをアプリケーションに追加しました。この手順では、フォトストリームから画像を表示するコードを追加します。

プログラムでコントロールにデータを追加するには、次の手順に従います。

1. [**ページ**]を選択して[**プロパティ**]ウィンドウに移動し、稲妻の[**イベント**]ボタンをクリックしてイベントを表示します。次に、下にスクロールして、**Loaded** イベントの横にある領域をダブルクリックします。
   これで、コードエディタが開き、**Page_Loaded** イベントが追加されます。
2. 次の **imports** 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
Imports System
Imports System.Collections.Generic
Imports System.Linq
Imports System.Net
Imports System.Xml.Linq
Imports Windows.UI.Popups
Imports Windows.UI.Xaml
Imports Windows.UI.Xaml.Controls
```

C# コードの書き方

```
C#
```


Copyright © GrapeCity, Inc. All rights reserved.

```
using C1.Xaml;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Xml.Linq;
using Windows.UI.Popups;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
```

3. **Page_Loaded** イベントハンドラ内に次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
LoadVideos()
```

C# コードの書き方

```
C#
LoadVideos();
```

4. **MainPage** クラス内の **Page_Loaded** イベントの下に次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
Private Async Function LoadVideos() As Task
Dim youtubeUrl = "https://gdata.youtube.com/feeds/api/videos?q=windows+8&max-results=50"
Dim AtomNS = "http://www.w3.org/2005/Atom"
Dim MediaNS = "http://search.yahoo.com/mrss/"

Dim videos = New List(Of Video)()
Dim client = WebRequest.CreateHttp(New Uri(youtubeUrl))
Dim response = Await client.GetResponseAsync()

Try
  #Region "** parse you tube data"
  Dim doc = XDocument.Load(response.GetResponseStream())
  For Each entry As var In doc.Descendants(XName.[Get]("entry", AtomNS))
      Dim title = entry.Element(XName.[Get]("title", AtomNS)).Value
      Dim thumbnail = ""
      Dim group = entry.Element(XName.[Get]("group", MediaNS))
      Dim thumbnails = group.Elements(XName.[Get]("thumbnail", MediaNS))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        Dim thumbnailElem = thumbnails.FirstOrDefault()
        If thumbnailElem IsNot Nothing Then
            thumbnail = thumbnailElem.Attribute("url").Value
        End If
        Dim alternate = entry.Elements(XName.[Get]("link", AtomNS)).Where(Function(elem)
elem.Attribute("rel").Value = "alternate").FirstOrDefault()
        Dim link = alternate.Attribute("href").Value
        videos.Add(New Video() With { _
            Key .Title = title, _
            Key .Link = link, _
            Key .Thumbnail = thumbnail _
        })
    Next
    #End Region

    tileListBox.ItemsSource = videos
    loading.Visibility = Visibility.Collapsed
    tileListBox.Visibility = Visibility.Visible
Catch
    Dim dialog = New MessageDialog("There was an error when attempting to download data from you tube.")
    dialog.ShowAsync()
End Try
End Function
```

C# コードの書き方

C#

```
private async void LoadVideos()
      {
          var youtubeUrl = "https://gdata.youtube.com/feeds/api/videos?q=windows+8&max-results=50";
          var AtomNS = "http://www.w3.org/2005/Atom";
          var MediaNS = "http://search.yahoo.com/mrss/";

          var videos = new List<Video>();
          var client = WebRequest.CreateHttp(new Uri(youtubeUrl));
          var response = await client.GetResponseAsync();

          try
          {
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            #region ** parse you tube data
            var doc = XDocument.Load(response.GetResponseStream());
            foreach (var entry in doc.Descendants(XName.Get("entry", AtomNS)))
            {
                var title = entry.Element(XName.Get("title", AtomNS)).Value;
                var thumbnail = "";
                var group = entry.Element(XName.Get("group", MediaNS));
                var thumbnails = group.Elements(XName.Get("thumbnail", MediaNS));
                var thumbnailElem = thumbnails.FirstOrDefault();
                if (thumbnailElem != null)
                    thumbnail = thumbnailElem.Attribute("url").Value;
                var alternate = entry.Elements(XName.Get("link", AtomNS)).Where(elem =>
elem.Attribute("rel").Value == "alternate").FirstOrDefault();
                var link = alternate.Attribute("href").Value;
                videos.Add(new Video() { Title = title, Link = link, Thumbnail = thumbnail });
            }
            #endregion

            tileListBox.ItemsSource = videos;
            loading.Visibility = Visibility.Collapsed;
            tileListBox.Visibility = Visibility.Visible;
        }
        catch
        {
            var dialog = new MessageDialog("There was an error when attempting to download data from you
tube.");
            dialog.ShowAsync();
        }
}
```

5. 上のコードは、YouTube から画像を取得し、**C1TileListBox** をビデオのリストに連結します。
6. **MainPage** クラス内で、追加したコードの下に次のコードを追加します。

Visual Basic コードの書き方

Visual Basic

```
Private Sub tileListBox_ItemClick(sender As Object, e As EventArgs)
  Dim video = TryCast(TryCast(sender, C1ListBoxItem).Content, Video)
  NavigateUrl(video.Link)
End Sub
#Region "** public properties"
```


Copyright © GrapeCity, Inc. All rights reserved.

```
Public Property Orientation() As Orientation
  Get
      Return tileListBox.Orientation
  End Get
  Set
      tileListBox.Orientation = value
  End Set
End Property

Public Property ItemWidth() As Double
  Get
      Return tileListBox.ItemWidth
  End Get
  Set
      tileListBox.ItemWidth = value
  End Set
End Property

Public Property ItemHeight() As Double
  Get
      Return tileListBox.ItemHeight
  End Get
  Set
      tileListBox.ItemHeight = value
  End Set
End Property
```

C# コードの書き方

C#

```
private void tileListBox_ItemClick(object sender, EventArgs e)
      {
         var video = (sender as C1ListBoxItem).Content as Video;
         NavigateUrl(video.Link);
      }



      #region ** public properties
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        public Orientation Orientation
        {
            get
            {
                return tileListBox.Orientation;
            }
            set
            {
                tileListBox.Orientation = value;
            }
        }

        public double ItemWidth
        {
            get
            {
                return tileListBox.ItemWidth;
            }
            set
            {
                tileListBox.ItemWidth = value;
            }
        }

        public double ItemHeight
        {
            get
            {
                return tileListBox.ItemHeight;
            }
            set
            {
                tileListBox.ItemHeight = value;
            }
    }
```

7. **MainPage** クラスの直後に次のコードを追加します。

Visual Basic コードの書き方

Copyright © GrapeCity, Inc. All rights reserved.

Visual Basic

```
Public Class Video
  Public Property Title() As String
      Get
          Return m_Title
      End Get
      Set
          m_Title = Value
      End Set
  End Property
  Private m_Title As String
  Public Property Thumbnail() As String
      Get
          Return m_Thumbnail
      End Get
      Set
          m_Thumbnail = Value
      End Set
  End Property
  Private m_Thumbnail As String
  Public Property Link() As String
      Get
          Return m_Link
      End Get
      Set
          m_Link = Value
      End Set
  End Property
  Private m_Link As String
End Class
```

C# コードの書き方

C#

```
public class Video
  {
      public string Title { get; set; }
      public string Thumbnail { get; set; }
      public string Link { get; set; }
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
```

ここまでの成果

これで、**C1TileTileListBox** にデータを追加できました。次の「手順3：TileListBox アプリケーションの実行」では、**TileListBox for UWP** の機能について説明します。

手順3：TileListBox アプリケーションの実行

これまでに、UWP スタイルアプリケーションを作成し、外観と動作をカスタマイズしたので、次にアプリケーションを実行します。アプリケーションを実行し、**TileListBox 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. [**デバッグ**]メニューから[**デバッグ開始**]を選択し、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。アプリケーションが表示され、画像が表示されます。
2. コントロールの右側にあるスクロールバーを使用して、ビデオ画像をスクロールします。

タッチ機能がある場合は、ピンチして画像をズームしてみてください。

ここまでの成果

おめでとうございます。これで **TileListBox for UWP** クイックスタートは完了です。**C1TileListBox** コントロールを使用するアプリケーションを作成し、アプリケーションの実行時機能をいくつか確認することができました。

重要なヒント

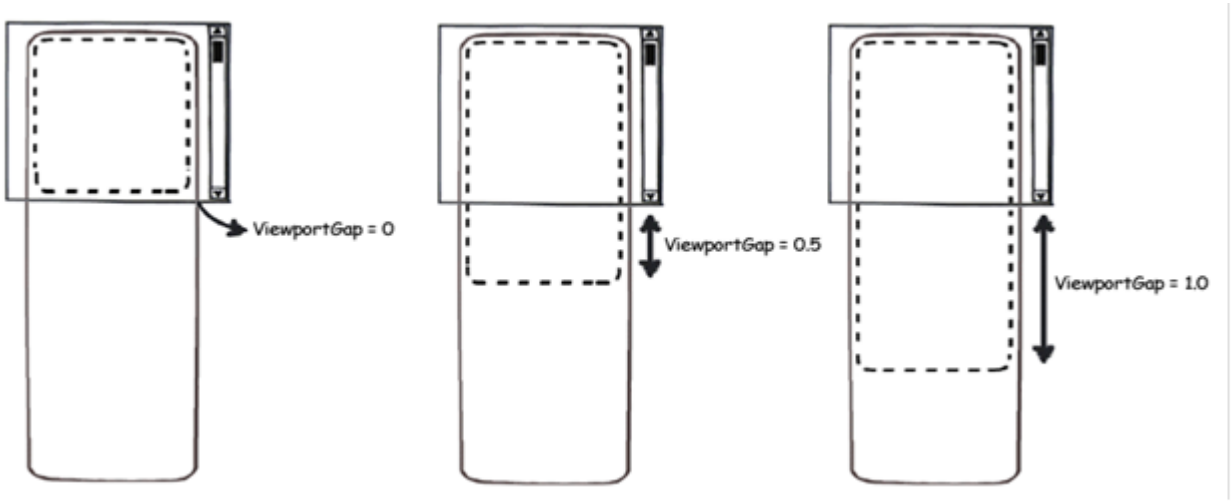
以下のヒントを活用することで、**ListBox** のコントロールを使用する際に、パフォーマンスを最大限に高めることができます。

- **PreviewTemplate の使用** – レイアウトによる負荷が高まることを避けるには、**PreviewTemplate** を使用して、ズームアウト中や高速スクロール中などのプレビュー状態のときにサイズの小さなテンプレートをレンダリングします。たとえば、**PreviewTemplate** でサムネイル画像を表示し、完全な **ItemTemplate** で大きな画像を表示することができます。

```
<Xaml:C1ListBox x:Name="listBox"
                ItemsSource="{Binding}"
                RefreshWhileScrolling="False">
  <Xaml:C1ListBox.PreviewItemTemplate>
    <DataTemplate>
      <Grid Background="Gray">
        <Image Source="{Binding Thumbnail}" Stretch="UniformToFill"/>
      </Grid>
    </DataTemplate>
  </Xaml:C1ListBox.PreviewItemTemplate>
  <Xaml:C1ListBox.ItemTemplate>
    <DataTemplate>
      <Grid>
        <Image Source="{Binding Content}" Stretch="UniformToFill"/>
      </Grid>
    </DataTemplate>
  </Xaml:C1ListBox.ItemTemplate>
</Xaml:C1ListBox>
```


Copyright © GrapeCity, Inc. All rights reserved.

- **ViewportGap プロパティと ViewportPreviewGap プロパティの調整** – これらの係数値は、あらかじめレンダリングするビューポート外の項目のサイズを決定します。値が大きくなるほど、画面外の項目が現れてレンダリングされる速度は上がりますが、毎回コントロールがレイアウトパスを受け取るためにかかる時間は長くなります。たとえば、これを 0.5 に設定した場合は、ビューポートが元のビューポートの両側に合わせて画面の半分だけ大きくなるように拡大されます。



- **RefreshWhileScrolling を False に設定** – スクロール中にビューポートを更新するかどうかを決定します。False に設定した場合は、ユーザーが高速にスクロール操作すると、項目が停止してレンダリングできるようになるまで、項目は空白で表示されるか、「読み込み中」と表示されます。

C1ListBox の使い方

ListBox for UWP には、**C1ListBox** および **C1TileListBox** コントロールが含まれています。**C1ListBox** は標準の **ListBox** コントロールに似ていますが、ズームなどの機能が追加されています。**C1TileListBox** を使用すると、項目をタイル表示したリストを作成することができます。**C1ListBox** コントロールや **C1TileListBox** コントロールを使用して、リストをタイル状にレイアウトして表示したり、光学ズーム付きで表示することができます。

基本的なプロパティ

ListBox 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。

次のプロパティを使用して、**C1ListBox** コントロールをカスタマイズできます。

プロパティ	説明
ActualMaxZoom	実際の最大ズームを取得します。
ActualMinZoom	実際の最小ズームを取得します。
ActualZoom	実際のズームを取得します。
IsScrolling	このリストがスクロール中であるかどうかを示す値を取得します。
IsZooming	このリストがズーム中であるかどうかを示す値を取得します。
ItemHeight	各項目の高さを取得または設定します。
Items	コントロールのコンテンツを生成するために使用されるコレクションを取得します。(ItemsControl を継承します。)
ItemsPanel	項目のレイアウトを制御するパネルを定義するテンプレートを取得または設定します。(ItemsControl を継承します。)

Copyright © GrapeCity, Inc. All rights reserved.

ItemsSource	ItemsControl のコンテンツを生成するために使用されるコレクションを取得または設定します。 (ItemsControl を継承します。)
ItemTemplate	リストのすべての項目に適用されるテンプレートです。 (C1ItemsControl を継承します。)
ItemTemplateSelector	同じ型の項目に適用する複数のテンプレートを指定するために使用されるテンプレートセレクタです。 (C1ItemsControl を継承します。)
ItemWidth	各項目の幅を取得または設定します。
MaxZoom	使用できる最大ズームを取得または設定します。
MinZoom	使用できる最小ズームを取得または設定します。
Orientation	リストを表示する方向を取得または設定します。
Panel	この項目コントロールに関連付けられているパネルを取得します。
PreviewItemTemplate	項目のプレビューに使用されるテンプレートを取得または設定します。
RefreshWhileScrolling	スクロールの実行中にビューポイントを更新する必要があるかどうかを示す値を取得または設定します。
ScrollViewer	この項目コントロールに属するスクロールビューアテンプレートパーツを取得します。
ViewportGap	毎回のレイアウトパスでビューポートのサイズを決定する係数を取得または設定します。0を指定すると、ビューポートのサイズはスクロールビューアのビューポートと同じになります。0.5を指定すると、ビューポートが元のビューポートの両側に合わせて画面の半分だけ大きくなるように拡大されます。
ViewportPreviewGap	毎回のレイアウトパスでプレビューモードの項目をレンダリングするためのビューポートのサイズを決定する係数を取得または設定します。
Zoom	このリストに適用されるズームを取得または設定します。
ZoomMode	ズームが有効か無効かを取得または設定します。

次のプロパティを使用して、**C1ListBoxItem** をカスタマイズできます。

プロパティ	説明
PreviewContent	プレビュー状態のコンテンツを取得または設定します。
PreviewContentTemplate	プレビュー状態の場合に使用する DataTemplate を取得または設定します。
State	項目の状態（Preview または Full）を取得または設定します。

光学ズーム

ListBox for UWP コントロールは、光学ズーム機能をサポートしています。ユーザーは、ピンチジェスチャを使用して、項目のサイズ変更操作を直観的に行うことができます。ズーム動作は流れるようにスムースです。アプリケーションのパフォーマンスが犠牲になることはありません。

ZoomMode プロパティと **Zoom** プロパティを使用して、ズームをカスタマイズできます。**ZoomMode** プロパティは、ズームが有効か無効かを取得または設定します。**Zoom** プロパティは、コントロールに適用されるズーム値を取得または設定します。**ZoomChanged** イベントは、コントロールのズーム値が変更されたときにトリガされます。

Copyright © GrapeCity, Inc. All rights reserved.

UI の仮想化

ListBox コントロールは UI の仮想化をサポートしています。そのため、動作は極めて高速であり、数千個の項目を表示してもパフォーマンスはほとんど低下しません。各レイアウトパスにレンダリングする項目の数を決定するには、**ViewportGap** プロパティと **ViewportPreviewGap** プロパティを設定します。これらのプロパティをシナリオに応じて調整できます。

ViewportGap プロパティは、毎回のレイアウトパスでビューポートのサイズを決定する係数を取得または設定します。0を指定すると、ビューポートのサイズはスクロールビューアのビューポートと同じになります。0.5 を指定すると、ビューポートが元のビューポートの両側に合わせて画面の半分だけ大きくなるように拡大されます。

ViewportPreviewGap プロパティは、毎回のレイアウトパスでプレビューモードの項目をレンダリングするためのビューポートのサイズを決定する係数を取得または設定します。

向き

ListBox コントロールは、水平方向と垂直方向の両方をサポートしています。このため、より多くのレイアウトシナリオに対応することができます。コントロールの方向を設定するには、**Orientation** プロパティを **Horizontal** または **Vertical** に設定します。

プレビュー状態

パフォーマンスを可能な限り高めるために、**ListBox** コントロールは、ビューポートの外の項目をプレビュー状態でレンダリングすることができます。標準の **ItemTemplate** と同様に、**Preview** テンプレートは、ズームアウト中または高速スクロール中などのプレビュー状態にある項目の外観を定義します。その後、項目のスクロールやズームが終了すると、完全な項目テンプレートに切り替えられます。

Input for UWP

電話番号、郵便番号、パーセンテージなどの入力をスマートに行います。**Input for UWP** は、マスク付き入力用のコントロールと数値入力用のコントロールの2つを提供します。書式設定されたテキストを自動的に表示すると共に、有効な入力をすばやく収集できます。

Input for UWP には、次のコントロールがあります。

- **NumericBox for UWP**
  NumericBox for UWP は、通貨やパーセンテージなどの書式設定された数値を表示および編集するためのテキストボックスを提供します。このコントロールは、スマート入力とインクリメントボタンを備えています。
- **MaskedTextBox for UWP**
  MaskedTextBox for UWP は、入力マスクによる検証機能を提供します。**C1MaskedTextBox** コントロールは、ユーザーが無効な文字を入力できないようにするためのマスクをテキストボックスに提供します。

次のトピックに従って、**Input for UWP** を簡単に使用できます。

主な特長

Input 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。**Input for UWP** は、次の主な特長を備えています。

- **標準の書式文字列**
  Mask または Format プロパティを使用して、ユーザー入力後、即座に書式設定が行われます。C1MaskedTextBox および C1NumericBox コントロールは、Microsoft によって定義されている標準の書式設定文字列をサポートし、従来の Windows フォームコントロールと同じ構文を使用します。Format プロパティでは、使い慣れた .NET 書式文字列を使用して、小数点位置の指定を含むさまざまな書式で数値を表示できます。サポートされている書式は、固定小数点

Copyright © GrapeCity, Inc. All rights reserved.

(F)、数値(N)、汎用(G)、通貨(C)、指数(E)、16 進数(X)、およびパーセント(P)です。

- **プロンプトとリテラルの表示**
  1つのプロパティを設定するだけで、**C1MaskedTextBox** コントロールでプロンプト文字とリテラルを表示するかどうかを選択できます。プロンプト文字は、テキストを入力できることをユーザーに示します(_ や * など)。リテラルは、マスク内でそのまま表示される非マスク文字です(/ や - など)。
- **数値範囲**
  C1NumericBox コントロールでは、**Minimum** プロパティと **Maximum** プロパティを設定して入力を特定の数値範囲に制限することができます。
- **ウォーターマークのサポート**
  Watermark プロパティを使用して、どのような値を入力する必要があるかを説明する簡単なヒントをユーザーに提供できます。ウォーターマークは、テキストが入力されるまでコントロールに表示されます。

クイックスタート

このクイックスタートガイドは、**Input 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。

NumericBox クイックスタート

このクイックスタートでは、5つの C1NumericBox コントロールを含むアプリケーションを作成します。これらのコントロールはロックとして機能します。すべてのコントロールにそれぞれ正しい暗証番号が入力されると、コントロールはロックされて非アクティブになり、ユーザーを Web サイトに導くボタンが表示されます。

手順1:UWP スタイルのアプリケーションの作成

この手順では、最初に Visual Studio で **NumericBox for UWP** を使用する UWP スタイルのアプリケーションを作成します。C1NumericBox コントロールをアプリケーションに追加するだけで、完全な機能を備えた数値エディタとして使用できます。さらに、そのコントロールをアプリケーションに合わせてカスタマイズできます。

プロジェクトを設定するには、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、言語の下で[**Windows ストア**]を選択し、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. 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。
4. ツールボックスに移動し、[**StackPanel**]アイコンをダブルクリックして、**MainPage.xaml** にパネルを追加します。
5. `x:Name="sp1" Width="Auto" Height="Auto" Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center"` を `<StackPanel>` タグに追加します。次のようになります。

   ```
   <StackPanel x:Name="sp1" Width="Auto" Height="Auto" Orientation="Vertical"
   HorizontalAlignment="Center" VerticalAlignment="Center"></StackPanel>
   ```

   これで、パネル内の要素は、中央で縦方向に配置されて表示されます。
6. プロジェクトの **XAML** ウィンドウで、カーソルを `<StackPanel>` タグと `</StackPanel>` タグの間に置きます。
7. Visual Studio のツールボックスに移動し、標準の **TextBlock** コントロールをダブルクリックしてプロジェクトに追加します。
8. `x:Name="tb1" Text="組み合わせの入力" Margin="5" FontSize="24"` を `<TextBlock>` タグに追加して、**TextBlock** に名前を付け、コンテンツを追加します。次のようになります。

   ```
   <TextBlock x:Name="tb1" Text="組み合わせを入力してください" Margin="5" FontSize="24"/>
   ```


Copyright © GrapeCity, Inc. All rights reserved.

9. ツールボックスに移動し、[**StackPanel**]アイコンをダブルクリックして、**TextBlock** のすぐ下にある既存の **StackPanel** にパネルを追加します。
10. `x:Name="sp2" Width="Auto" Height="Auto" Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center"` を `<StackPanel>` タグに追加します。次のようになります。

```
<StackPanel x:Name="sp2" Width="Auto" Height="Auto" Orientation="Vertical"
HorizontalAlignment="Center" VerticalAlignment="Center"></StackPanel>
```

これで、パネル内の要素は、中央で横方向に配置されて表示されます。

11. カーソルを最初の `</StackPanel>` タグと2番目の `</StackPanel>` タグの間に置き、次のマークアップを追加して、2番目のラベルを作成します。

```
<TextBlock x:Name="tb2" Text="組み合わせが不正です" Foreground="Red" Margin="5" FontSize="18"/>
```

12. カーソルを `<TextBlock>` タグと2番目の `</StackPanel>` タグの間に置き、次のマークアップを追加して、非表示のボタンを作成します。

```
<Button x:Name="btn1" Content="Enter" Height="60" Visibility="Collapsed" Click="btn1_Click" Margin="5">
</Button>
```

btn1_Click イベントハンドラは、後でコードに追加します。

これで、UWP スタイルのアプリケーションを作成し、アプリケーションのユーザーインタフェースを設定し、アプリケーションにコントロールを追加できました。次の手順では、C1NumericBox コントロールを追加して、アプリケーションの設定を完成させます。

手順2：C1NumericBox のコントロールの追加

前の手順では、新しい UWP スタイルのプロジェクトを作成し、アプリケーションに5つのコントロールを追加しました。この手順では、引き続き、C1NumericBox コントロールを追加してアプリケーションをカスタマイズします。

次の手順に従います。

1. プロジェクトの XAML ウィンドウで、カーソルを `<StackPanel x:Name="sp2">` タグと `</StackPanel>` タグの間に置きます。
2. ツールボックスに移動し、[**C1NumericBox**]アイコンをダブルクリックして、**StackPanel** にコントロールを追加します。XAML マークアップは次のようになります。

```
<Xaml:C1NumericBox></Xaml:C1NumericBox>
```

C1.Xaml 名前空間と `<Xaml:C1NumericBox></Xaml:C1NumericBox>` タグがプロジェクトに追加されています。

3. `x:Name="c1nb1"` を `<Xaml:C1NumericBox>` タグに追加して、コントロールに名前を付けます。次のようになります。

```
<Xaml:C1NumericBox x:Name="c1nb1">
```

それに一意の識別子を付けると、コードでそのコントロールにアクセスできるようになります。

4. `<Xaml:C1NumericBox>` タグに `Margin="5"` を追加して、マージンを追加します。次のようになります。

```
<Xaml:C1NumericBox x:Name="c1nb1" Margin="2">
```

ページ上でコントロールが間隔を開けて表示されます。

Copyright © GrapeCity, Inc. All rights reserved.

5. `Minimum="0" Maximum="9"` を `<Xaml:C1NumericBox>` タグに追加して、コントロールに制限を設定します。次のようになります。

```
<Xaml:C1NumericBox x:Name="c1nb1" Margin="2" Minimum="0" Maximum="9">
```

Minimum プロパティと Maximum プロパティは、コントロールに入力できる最小値と最大値を設定します。データ検証機能が組み込まれているため、ユーザーはこの範囲外の値を入力できなくなります。

6. `<Xaml:C1NumericBox>` タグに `ValueChanged="c1nb1_ValueChanged"` を追加します。次のようになります。

```
<Xaml:C1NumericBox x:Name="c1nb1" Margin="2" Minimum="0" Maximum="9"
ValueChanged="c1nb1_ValueChanged">
```

c1nb1_ValueChanged イベントハンドラのコードは、この後の手順で追加します。

7. 既存の `<Xaml:C1NumericBox x:Name="c1nb1"></Xaml:C1NumericBox>` タグのすぐ下に、次の XAML を追加します。

```
<Xaml:C1NumericBox x:Name="c1nb2" Minimum="0" Maximum="9" Margin="5"
ValueChanged="c1nb2_ValueChanged"></Xaml:C1NumericBox>
<Xaml:C1NumericBox x:Name="c1nb3" Minimum="0" Maximum="9" Margin="5"
ValueChanged="c1nb3_ValueChanged"></Xaml:C1NumericBox>
<Xaml:C1NumericBox x:Name="c1nb4" Minimum="0" Maximum="9" Margin="5"
ValueChanged="c1nb4_ValueChanged"></Xaml:C1NumericBox>
<Xaml:C1NumericBox x:Name="c1nb5" Minimum="0" Maximum="9" Margin="5"
ValueChanged="c1nb5_ValueChanged"></Xaml:C1NumericBox>
```

これで、さらに4つの C1NumericBox コントロールが追加され、合計5つのコントロールがページに置かれます。

これで、アプリケーションに **C1NumericBox** コントロールを追加し、これらのコントロールをカスタマイズできました。次の手順では、アプリケーションにコードを追加します。

手順3：アプリケーションへのコードの追加

これまでの手順では、アプリケーションのユーザーインタフェースを設定し、C1NumericBox、**TextBlock**、**Button** の各コントロールをアプリケーションに追加しました。この手順では、アプリケーションにコードを追加して完成させます。

次の手順に従います。

1. [**表示**]→[**コード**]を選択してコードビューに切り替えます。
2. 次の imports 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 Windows.UI.Xaml.Media
Imports Windows.UI.Xaml.Navigation
Imports Windows.UI
Imports C1.Xaml
```

C# コードの書き方

```
C#
using Windows.UI.Xaml.Media;
```


Copyright © GrapeCity, Inc. All rights reserved.

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

3. **MainPage** クラスのすぐ内側にある次のグローバル変数を初期化します。

Visual Basic コードの書き方

```
Visual Basic
Dim nb1 As Integer = 5
Dim nb2 As Integer = 2
Dim nb3 As Integer = 3
Dim nb4 As Integer = 7
Dim nb5 As Integer = 9
```

C# コードの書き方

```
C#
int nb1 = 5;
int nb2 = 2;
int nb3 = 3;
int nb4 = 7;
int nb5 = 9;
```

これらの数字は、正しい「暗証番号」としてアプリケーションで使用されます。ユーザーが実行時に正しい番号の組み合わせを入力すると、ボタンが表示されます。

4. **Button1_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 btn1_Click(ByVal sender As System.Object, ByVal e As System.Windows.RoutedEventArgs)
Handles btn1.Click
    DefaultLaunch()
End Sub
```

C# コードの書き方

```
C#
private void btn1_Click(object sender, RoutedEventArgs e)
{
    DefaultLaunch();
}
```

5. **Button1_Click** イベントハンドラのすぐ下に次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
async Sub DefaultLaunch()
  ' 起動する URI
  Dim uri As New Uri("c1.grapecity.com")
  ' URI を起動します
  Dim success = await Windows.System.Launcher.LaunchUriAsync(uri)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
  If success Then
    ' URI が起動されました
  Else
    ' URI の起動に失敗しました
  End If
End Sub
```

C# コードの書き方

```
C#
async void DefaultLaunch()
{
  // 起動する URI
  string uriToLaunch = @"http://c1.grapecity.com/";
  var uri = new Uri(uriToLaunch);
  // URI を起動します
  var success = await Windows.System.Launcher.LaunchUriAsync(uri);
  if (success)
  {
    // URI が起動されました
  }
  else
  {
    // URI の起動に失敗しました
  }
}
```

実行時にボタンを押すと、ComponentOne Web サイトが開きます。

6. 次に、以下のカスタム **NBValidation** イベントをコードに追加します。

Visual Basic コードの書き方

```
Visual Basic
Private Sub NBValidation()
  If Me.c1nb1.Value = nb1 And Me.c1nb2.Value = nb2 And Me.c1nb3.Value = nb3 And Me.c1nb4.Value =
nb4 And Me.c1nb5.Value = nb5 Then
    Me.tb2.Foreground = New SolidColorBrush(Colors.Green)
    Me.tb2.Text = "組み合わせが正しい"
    Me.c1nb1.IsReadOnly = True
    Me.c1nb2.IsReadOnly = True
    Me.c1nb3.IsReadOnly = True
    Me.c1nb4.IsReadOnly = True
    Me.c1nb5.IsReadOnly = True
    Me.btn1.Visibility = Visibility.Visible
  End If
End Sub
```

C# コードの書き方

```
C#
private void NBValidation()
{
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    if (this.c1nb1.Value == nb1 & this.c1nb2.Value == nb2 & this.c1nb3.Value == nb3 & this.c1nb4.Value ==
nb4 & this.c1nb5.Value == nb5)
    {
        this.tb2.Foreground = new SolidColorBrush(Colors.Green);
        this.tb2.Text = "組み合わせが正しい";
        this.c1nb1.IsReadOnly = true;
        this.c1nb2.IsReadOnly = true;
        this.c1nb3.IsReadOnly = true;
        this.c1nb4.IsReadOnly = true;
        this.c1nb5.IsReadOnly = true;
        this.btn1.Visibility = Visibility.Visible;
    }
}
```

ユーザーが（上の手順3で示した）正しい番号を入力すると、C1NumericBox コントロールは読み取り専用に設定されて編集できなくなります。コントロールの下にあるラベルのテキストは、正しいコードが入力されていることを示すように変更されます。また、ComponentOne Web サイトに移動するためのボタンが表示されます。

7. **NBValidation** を初期化するための **C1NumericBox_ValueChanged** イベントハンドラを追加します。コードは次のようになります。

Visual Basic コードの書き方

Visual Basic

```
Private Sub c1nb1_ValueChanged(ByVal sender As System.Object, ByVal e As
C1.Xaml.PropertyChangedEventArgs(Of System.Double)) Handles c1nb1.ValueChanged
    NBValidation()
End Sub
Private Sub c1nb2_ValueChanged(ByVal sender As System.Object, ByVal e As
C1.Xaml.PropertyChangedEventArgs(Of System.Double)) Handles c1nb2.ValueChanged
    NBValidation()
End Sub
Private Sub c1nb3_ValueChanged(ByVal sender As System.Object, ByVal e As
C1.Xaml.PropertyChangedEventArgs(Of System.Double)) Handles c1nb3.ValueChanged
    NBValidation()
End Sub
Private Sub c1nb4_ValueChanged(ByVal sender As System.Object, ByVal e As
C1.Xaml.PropertyChangedEventArgs(Of System.Double)) Handles c1nb4.ValueChanged
    NBValidation()
End Sub
Private Sub c1nb5_ValueChanged(ByVal sender As System.Object, ByVal e As
C1.Xaml.PropertyChangedEventArgs(Of System.Double)) Handles c1nb5.ValueChanged
    NBValidation()
End Sub
```

C# コードの書き方

C#

```
private void c1nb1_ValueChanged(object sender, PropertyChangedEventArgs<double> e)
{
    NBValidation();
}
private void c1nb2_ValueChanged(object sender, PropertyChangedEventArgs<double> e)
{
```


Copyright © GrapeCity, Inc. All rights reserved.

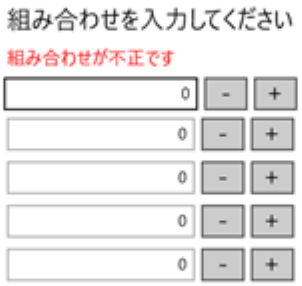
```
    NBValidation();
}
private void c1nb3_ValueChanged(object sender, PropertyChangedEventArgs<double> e)
{
    NBValidation();
}
private void c1nb4_ValueChanged(object sender, PropertyChangedEventArgs<double> e)
{
    NBValidation();
}
private void c1nb5_ValueChanged(object sender, PropertyChangedEventArgs<double> e)
{
    NBValidation();
}
```

この手順では、アプリケーションにコードを追加しました。次の手順では、アプリケーションを実行し、実行時の操作を確認します。

手順4：アプリケーションの実行

これまでに UWP スタイルのアプリケーションを作成し、外観と動作をカスタマイズしたので、次にアプリケーションを実行します。アプリケーションを実行し、**NumericBox 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. [**デバッグ**]メニューから[**デバッグ開始**]を選択し、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。



2. 最初の **C1NumericBox** コントロールに**5**が表示されるまで、このコントロールの「+」ボタンをクリックします。ボタンをクリックするたびに数字が1だけインクリメントされます。これは、Increment プロパティがデフォルトで**1**に設定されているためです。
3. 2番目の **C1NumericBox** の内部をクリックすると、「0」値が強調表示されます。「2」を入力して数字を書き換えます。
4. 3番目の **C1NumericBox** コントロールで「-」ボタンをクリックしてみると、数字が変わらないことがわかります。これは、Minimum プロパティが**0**に設定されているためです。コントロールは0未満の値を受け付けません。**3**が表示され

Copyright © GrapeCity, Inc. All rights reserved.

るまで「+」ボタンをクリックします。
5. 4番目の **C1NumericBox** コントロールで、**0**の前にカーソルを置き、クリックします。「5」を入力すると、「50」と表示されます。
6. 最後の **C1NumericBox** コントロールの内部をクリックします。4番目の **C1NumericBox** の **50** が**9**にリセットされることがわかります。これは、Maximum プロパティが**9**に設定されており、コントロールが9より大きい値を受け付けないためです。
7. 最後の **C1NumericBox** コントロールに**9**を入力します。
8. 4番目の **C1NumericBox** コントロールの「-」ボタンを2回クリックすると、**7**が表示されます。2番目の Label のテキストが変化し、ボタンが表示されます。
9. **C1NumericBox** コントロールへの入力を試みたり、「+」ボタンや「-」ボタンをクリックしても、入力は受け付けられません。これは、すべて正しい暗証番号を入力したときに IsReadOnly プロパティが **True** に設定され、コントロールの編集がロックされたためです。
10. 表示された[**Enter**]ボタンをクリックすると、ComponentOne Web サイトに移動します。

おめでとうございます。これで、**NumericBox** クイックスタートガイドは完了です。**NumericBox** アプリケーションを作成し、コントロールの外観と動作をカスタマイズし、アプリケーションの実行時機能をいくつか確認することができました。

MaskedTextBox クイックスタート

このクイックスタートでは、Visual Studio に新しいアプリケーションを作成し、MaskedTextBox コントロールを追加するほか、アプリケーション内のコントロールの外観や動作をカスタマイズします。

手順1：アプリケーションの設定

この手順では、最初に Visual Studio で **MaskedTextBox for UWP** を使用する UWP スタイルのアプリケーションを作成します。C1MaskedTextBox コントロールをアプリケーションに追加するだけで、完全な機能を備えた入力エディタとして使用できます。さらに、そのコントロールをアプリケーションに合わせてカスタマイズできます。

プロジェクトをセットアップし、C1MaskedTextBox 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、言語の下で[**Windows ストア**]を選択し、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. **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。
4. ツールボックスに移動し、**StackPanel** アイコンをダブルクリックしてページに追加します。
5. `x:Name="sp1" Width="Auto" Height="Auto" Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center"` を `<StackPanel>` タグに追加します。次のようになります。

```
<StackPanel x:Name="sp1" Width="Auto" Height="Auto" Orientation="Vertical"
HorizontalAlignment="Center" VerticalAlignment="Center"></StackPanel>
```

これで、パネル内の要素は、中央で縦方向に配置されて表示されます。

これで、UWP スタイルのアプリケーションを作成できました。次の手順では、**TextBlock** コントロールと C1MakedTextBox コントロールを追加してカスタマイズし、アプリケーションの設定を完成させます。

手順2：アプリケーションのカスタマイズ

前の手順では、新しい UWP スタイルのプロジェクトを作成し、アプリケーションに1つの **StackPanel** を追加しました。この手順では、引き続き、**TextBlock** コントロールと **C1MaskedTextBox** コントロールを追加してカスタマイズします。

Copyright © GrapeCity, Inc. All rights reserved.

次の手順に従います。

1. プロジェクトの XAML ウィンドウで、カーソルを `<StackPanel x:Name="sp1">` タグと `</StackPanel>` タグの間に置きます。
2. StackPanel 内に次のマークアップを追加して、2つの標準 TextBlock コントロールを追加します。

```
<TextBlock Margin="2,2,2,10" Name="tb1" Text="社員情報" />
<TextBlock FontSize="16" Margin="2,2,2,0" Text="社員 ID" />
```

3. 追加したマークアップのすぐ下にカーソルを置き、ツールボックスに移動します。**C1MaskedTextBox** アイコンをダブルクリックして、このコントロールを **StackPanel** 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。

```
<Xaml:C1MaskedTextBox x:Name="c1MaskedTextBox" Text="C1MaskedTextBox"/>
```

4. グリッド内で `Name="c1mtb1" VerticalAlignment="Top" Margin="2" Mask="000-00-0000" TextChanged="c1mtb1_TextChanged"` を `<Xaml:C1MaskedTextBox/>` タグに追加して、**C1MaskedTextBox** コントロールを初期化し、コントロールに名前を付けます。次のようになります。

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
    <Xaml:C1MaskedTextBox Name="c1mtb1" VerticalAlignment="Top" Margin="2" Mask="000-00-0000"
TextChanged="c1mtb1_TextChanged"/>
</Grid>
```

このマークアップでは、コントロールに名前を付け、マージンと配置を設定し、ボックスのコンテンツのマスクを設定しています。後の手順で、イベントハンドラのコードを追加します。

5. `<Xaml:C1MaskedTextBox>` タグの直後にカーソルを置き、次の XAML を追加して、**C1MaskedTextBox** および **TextBlock** コントロールを **StackPanel** に追加します。

```
<TextBlock x:Name="tb2" FontSize="16" Margin="2" />
<TextBlock FontSize="16" Margin="2,2,2,0" Text="名前"/>
<Xaml:C1MaskedTextBox Name="c1mtb2" VerticalAlignment="Top" Margin="2"
TextChanged="c1mtb2_TextChanged"></Xaml:C1MaskedTextBox>
<TextBlock x:Name="tb3" FontSize="16" Margin="2"/>
<TextBlock FontSize="16" Margin="2"  Text="入社日"/>
<Xaml:C1MaskedTextBox Name="c1mtb3" VerticalAlignment="Top" Margin="2" Mask="00/00/0000"
TextChanged="c1mtb3_TextChanged"></Xaml:C1MaskedTextBox>
<TextBlock x:Name="tb4" FontSize="16" Margin="2"/>
<TextBlock FontSize="16" Margin="2,2,2,0" Text="電話番号"/>
<Xaml:C1MaskedTextBox Name="c1mtb4" VerticalAlignment="Top" Margin="2" Mask="(999) 000-0000"
TextChanged="c1mtb4_TextChanged"></Xaml:C1MaskedTextBox>
<TextBlock x:Name="tb5" FontSize="16" Margin="2"/>
```

これで、アプリケーションのユーザーインタフェースを設定できました。次の手順では、コードをアプリケーションに追加します。

手順3：アプリケーションへのコードの追加

これまでの手順では、アプリケーションのユーザーインタフェースを設定し、いくつかのコントロールをアプリケーションに追加しました。この手順では、機能を追加するコードをアプリケーションに追加します。

次の手順に従います。

1. [**表示**]→[**コード**]を選択してコードビューに切り替えます。

Copyright © GrapeCity, Inc. All rights reserved.

2. コードビューで、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
```

C# コードの書き方

```
C#
using C1.Xaml;
```

3. 次の **C1MaskedTextBox_TextChanged** イベントハンドラをプロジェクトに追加します。

Visual Basic コードの書き方

```
Visual Basic
Private Sub c1mtb1_TextChanged(ByVal sender As System.Object, ByVal e As
System.Windows.Controls.TextChangedEventArgs) Handles c1mtb1.TextChanged
    Me.tb2.Text = "マスク: " & Me.c1mtb1.Mask & "  値: " & Me.c1mtb1.Value & "  テキスト: " &
Me.c1mtb1.Text
End Sub
Private Sub c1mtb2_TextChanged(ByVal sender As System.Object, ByVal e As
System.Windows.Controls.TextChangedEventArgs) Handles c1mtb2.TextChanged
    Me.tb3.Text = "マスク: " & Me.c1mtb2.Mask & "  値: " & Me.c1mtb2.Value & "  テキスト: " &
Me.c1mtb2.Text
End Sub
Private Sub c1mtb3_TextChanged(ByVal sender As System.Object, ByVal e As
System.Windows.Controls.TextChangedEventArgs) Handles c1mtb3.TextChanged
    Me.tb4.Text = "マスク: " & Me.c1mtb3.Mask & "  値: " & Me.c1mtb3.Value & "  テキスト: " &
Me.c1mtb3.Text
End Sub
Private Sub c1mtb4_TextChanged(ByVal sender As System.Object, ByVal e As
System.Windows.Controls.TextChangedEventArgs) Handles c1mtb4.TextChanged
    Me.tb5.Text = "マスク: " & Me.c1mtb4.Mask & "  値: " & Me.c1mtb4.Value & "  テキスト: " &
Me.c1mtb4.Text
End Sub
```

C# コードの書き方

```
C#
private void c1mtb1_TextChanged(object sender, TextChangedEventArgs e)
{
    this.tb2.Text = ": " + this.c1mtb1.Mask + " 値: " + this.c1mtb1.Value + " テキスト: " + this.c1mtb1.Text;
}
private void c1mtb2_TextChanged(object sender, TextChangedEventArgs e)
{
    this.tb3.Text = "マスク: " + this.c1mtb2.Mask + " 値: " + this.c1mtb2.Value + " テキスト: " + this.c1mtb2.Text;
}
private void c1mtb3_TextChanged(object sender, TextChangedEventArgs e)
{
    this.tb4.Text = "マスク: " + this.c1mtb3.Mask + " 値: " + this.c1mtb3.Value + " テキスト: " + this.c1mtb3.Text;
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
private void c1mtb4_TextChanged(object sender, TextChangedEventArgs e)
{
    this.tb5.Text = "マスク: " + this.c1mtb4.Mask + " 値: " + this.c1mtb4.Value + " テキスト: " + this.c1mtb4.Text;
}
```

4. 次のコードを Page コンストラクタに追加します。

Visual Basic コードの書き方

```
Visual Basic
Public Sub New()
    InitializeComponent()
    Me.c1mtb1_TextChanged(Nothing, Nothing)
    Me.c1mtb2_TextChanged(Nothing, Nothing)
    Me.c1mtb3_TextChanged(Nothing, Nothing)
    Me.c1mtb4_TextChanged(Nothing, Nothing)
End Sub
```

C# コードの書き方

```
C#
public MainPage()
{
    InitializeComponent();
    this.c1mtb1_TextChanged(null, null);
    this.c1mtb2_TextChanged(null, null);
    this.c1mtb3_TextChanged(null, null);
    this.c1mtb4_TextChanged(null, null);
}
```

この手順では、アプリケーションにコードを追加しました。次の手順では、アプリケーションを実行し、実行時の操作を確認します。

手順4：アプリケーションの実行

これまでに UWP スタイルのアプリケーションを作成し、外観と動作をカスタマイズしたので、次にアプリケーションを実行します。アプリケーションを実行し、**MaskedTextBox 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. [**デバッグ**]メニューから[**デバッグ開始**]を選択し、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。次のようになります。

Copyright © GrapeCity, Inc. All rights reserved.

2. 最初の **C1MaskedTextBox** コントロールに数字を入力します。
   コントロールの下にあるラベルには、マスク、現在の値、および現在のテキストが表示されます。
3. 2番目の **C1MaskedTextBox** コントロールに文字列を入力します。
   このコントロールにはマスクが設定されていません。したがって、必要なら、数字も他の文字もコントロールに入力できます。
4. 3番目の **C1MaskedTextBox** コントロールに文字列を入力してみます。入力できないことがわかります。Mask プロパティは、数字のみを受け付けるように設定されていました。代わりに数値を入力します。これは問題ありません。
5. 残りのコントロールに数字を入力します。
   各 C1MaskedTextBox コントロールの下に表示される Value プロパティには、リテラル文字が含まれません。一方、**Text** プロパティにはリテラル文字が含まれます。

おめでとうございます。これで、**MaskedTextBox** クイックスタートガイドは完了です。**MaskedTextBox for UWP** アプリケーションを作成し、コントロールの外観と動作をカスタマイズし、アプリケーションの実行時機能をいくつか確認することができました。

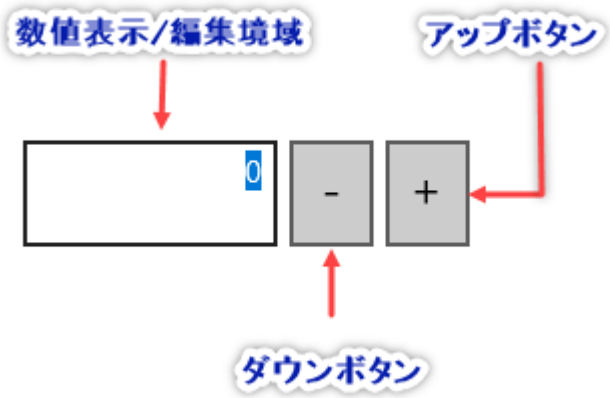
C1Input の使い方

以下のトピックでは、**C1NumericBox** コントロールと **C1MaskedTextBox** コントロールの機能について説明します。

C1NumericBox を操作する

NumericBox for UWP には **C1NumericBox** コントロールがあります。これは、数値の入力と編集を提供するシンプルなコントロールです。XAML ウィンドウに追加された C1NumericBox コントロールは、完全な機能を備えた数値エディタになります。デフォルトでは、コントロールのインタフェースは次の図のように表示されます。

Copyright © GrapeCity, Inc. All rights reserved.



コントロールは次の要素で構成されます。

- **アップ/ダウンボタン**
  ユーザーは、「+」(**アップ**)ボタンと「-」(**ダウン**)ボタンを使用して、コントロールに表示される値を変更できます。ボタンを押すたびに、**Increment** プロパティで指示された値(デフォルトでは1)だけ **Value** が変化します。デフォルトでは、、「+」ボタンと「-」ボタンが表示されます。これらのボタンを非表示にするには、**ShowButtons** プロパティを **False** に設定します。
- **数値表示/編集領域**
  現在の **Value** は、数値表示/編集領域に表示されます。ユーザーは、ボックスに入力して、Value プロパティを変更できます。デフォルトでは、ユーザーがこの数値を編集できます。コントロールの編集をロックするには、**IsReadOnly** を **True** に設定します。

数値書式設定

C1NumericBox コントロールに表示される数値の表示方法を変更できます。それには、**Format** プロパティを設定します。NumericBox for UWP は、Microsoft によって定義される標準の数値書式文字列をサポートします。詳細については、MSDN を参照してください。

Format 文字列は、書式を定義する英字または英字と数字の組み合わせで構成されます。デフォルトでは、Format プロパティは「F0」に設定されています。英字は書式タイプを指示します。この「F」は固定小数点を表し、数字は小数点以下の桁数を表します(この場合はなし)。

次の書式を使用できます。

書式指定子	名前	説明
C または c	**Currency**	数値は、金額を表す文字列に変換されます。変換は、現在のNumberFormatInfo オブジェクトの通貨書式情報によって制御されます。 精度指定子は、小数点以下の桁数を指示します。精度指定子を省略すると、現在の NumberFormatInfo オブジェクトで指定されるデフォルトの通貨精度が使用されます。
D または d	**Decimal**	この書式は、整数型でのみサポートされます。数値は 10 進数字(0～9)から成る文字列に変換されます。数字が負の場合は、先頭にマイナス記号が付きます。 精度指定子は、結果の文字列に必要な最小桁数を指示します。精度指定子によって指定された桁数になるように、必要に応じて、数字の左側がゼロで埋められます。 次の例は、Decimal 書式指定子で Int32 値を書式設定しています。
E または e	**Scientific(指数)**	数値は、「-d.ddd…E+ddd」または「-d.ddd…e+ddd」形式の文字列に変換されます。「d」はそれぞれ数字(0～9)を示します。数値が負である場合は、文字列の先頭にマイナス記号(-)が付加されます。小数点の前には、常に1桁の数字が置かれます。

Copyright © GrapeCity, Inc. All rights reserved.

		精度指定子は、小数点以下の桁数を指示します。精度指定子を省略すると、デフォルトで小数点以下6桁が使用されます。 書式指定子の大文字と小文字は、指数部分の前に「E」と「e」のどちらを付けるかを指示します。指数は、常にプラス記号（+）またはマイナス記号（-）と3桁以上の数字で表されます。この形式でない場合は、必要に応じて指数に0が付加されます。
F または f	**固定小数点**	数値は、「-ddd.ddd…」形式の文字列に変換されます。「d」はそれぞれ数字（0～9）を示します。数値が負である場合は、文字列の先頭にマイナス記号（-）が付加されます。 精度指定子は、小数点以下の桁数を指示します。精度指定子を省略すると、現在の NumberFormatInfo オブジェクトのNumberDecimalDigits プロパティによってデフォルトの数値精度が指定されます。
G または g	**General**	数値の型と精度指定子の有無に応じて、固定小数点または指数表記のいずれか簡単な方の形式に数値が変換されます。精度指定子を省略するか0にした場合は、次のリストに示すように、数値の型によってデフォルトの精度が決定されます。 • Byte または SByte：3 • Int16 または UInt16：5 • Int32 または UInt32：10 • Int64: 19 • UInt64: 20 • Single：7 • Double：15 • Decimal：29 数値を指数表記で表現する場合の指数部が -5 より大きく、精度指定子より小さい場合は、固定小数点表記が使用されます。それ以外の場合は、指数表記が使用されます。必要に応じて、結果には小数点が含まれ、末尾の0は省略されます。精度指定子があり、結果の有効桁数が指定された精度を超えている場合は、数値が丸められて末尾の余分な桁が除かれます。 上の規則の例外として、数値が Decimal で、精度指定子が省略されている場合があります。その場合は、常に固定小数点表記が使用され、末尾の0は保持されます。 指数表記を使用する場合は、書式指定子が「G」なら結果の指数部の前に「E」が付き、書式指定子が「g」なら「e」が付きます。指数部は少なくとも2桁あります。これは、「E」または「e」書式指定子によって生成される指数表記の書式（指数部は3桁以上）とは異なります。
N または n	**Number**	数値は、「-d,ddd,ddd.ddd…」形式の文字列に変換されます。「-」は負数の記号（必要に応じて）、「d」は数字（0～9）、「,」は数値の区切り記号、「.」は小数点記号を示します。実際の負数パターン、桁区切りの間隔、区切り記号、および小数点はそれぞれ、現在の NumberFormatInfo オブジェクトの NumberNegativePattern、NumberGroupSizes、NumberGroupSeparator、および NumberDecimalSeparator プロパティによって指定されます。 精度指定子は、小数点以下の桁数を指示します。精度指定子を省略すると、現在の NumberFormatInfo オブジェクトの NumberDecimalDigits プロパティによってデフォルトの数値精度が指定されます。
P または p	**Percent**	数値は、PercentNegativePattern プロパティ（数値が負の場合）または PercentPositivePattern プロパティ（数値が正の場合）で定義されたとおりに、パーセントを表す文字列に変換されます。変換された数値は 100 倍されて、パーセンテージで示されます。 精度指定子は、小数点以下の桁数を指示します。精度指定子を省略すると、現在の NumberFormatInfo オブジェクトで指定されるデフォルトの数値精度が使用されます。

Copyright © GrapeCity, Inc. All rights reserved.

R または r	**ラウンドトリップ**	この書式は、Single 型と Double 型でのみサポートされます。ラウンドトリップ指定子を指定すると、文字列に変換された数値が再度同じ数値に解析されます。この指定子を使用して書式設定された数値は、まず、Double 型の場合は 15 スペースの精度、Single 型の場合は7スペースの精度の汎用の書式でテストされます。値が再度正しく同じ数値に解析された場合は、汎用の書式指定子を使用して書式設定されます。ただし、値が再度同じ数値に解析されなかった場合は、Double 型の場合は 17 桁の精度、Single 型の場合は9桁の精度で書式設定されます。 精度指定子は指定してもかまいませんが、無視されます。この指定子を使用する場合は、精度よりラウンドトリップが優先されます。
X または x	**Hexadecimal**	この書式は、整数型でのみサポートされます。数値が 16 進数字の文字列に変換されます。書式指定子の大文字と小文字は、9より大きい 16 進数字に大文字を使用するか小文字を使用するかを指示します。たとえば、「X」を使用すると「ABCDEF」になり、「x」を使用すると「abcdef」になります。 精度指定子は、結果の文字列に必要な最小桁数を指示します。精度指定子によって指定された桁数になるように、必要に応じて、数字の左側がゼロで埋められます。
その他の文字	**(不明な指定子)**	(不明な指定子は実行時に FormatException を生成します)

入力検証

Minimum プロパティと **Maximum** プロパティを使用して、実行時の制限になる数値範囲を設定できます。Minimum プロパティや Maximum プロパティを設定すると、ユーザーは Minimum より大きい値や Maximum より小さい値を選択できなくなります。

Minimum および Maximum プロパティを設定する場合は、Minimum を Maximum より小さくする必要があります。また、Value プロパティには、Minimum と Maximum の範囲内の値を設定してください。

RangeValidationMode プロパティを使用して範囲検証モードを選択することもできます。このプロパティは、入力された数字をいつ検証するかを制御します。RangeValidationMode には、次のオプションの1つを設定できます。

オプション	説明
Always	このモードでは、ユーザーは範囲外の値を入力できません。
AlwaysTruncate	このモードでは、ユーザーは範囲外の値を入力できません。制限を超えた値は切り捨てられます。
OnLostFocus	このモードでは、コントロールがフォーカスを失ったときに値が切り捨てられます。

C1MaskTextBox の使い方

MaskedTextBox for UWP には、**C1MaskedTextBox** コントロールがあります。これは、入力を自動的に検証するマスクをテキストボックスに付けたシンプルなコントロールです。XAML ウィンドウに追加された C1MaskedTextBox コントロールは、完全な機能を備えたテキストボックスになり、マスクを使用してそれをさらにカスタマイズできます。

C1MaskedTextBox コントロールはテキストボックスのような外観で、カスタマイズできる基本的なテキスト入力領域があります。

マスクの書式設定

Mask プロパティを設定することで、入力検証を提供し、**C1MaskedTextBox** コントロールに表示される内容の表示方法を書式設定できます。MaskedTextBox for UWP は、Microsoft によって定義された標準の数値書式文字列をサポートし、**Mask** プロパティは、WinForms の標準の **MaskedTextBox** コントロールと同じ構文を使用します。これにより、アプリケーションやプラットフォーム間でマスクを再利用することが容易になります。

Copyright © GrapeCity, Inc. All rights reserved.

デフォルトでは、Mask プロパティは未設定で、入力マスクは適用されません。マスクを適用する場合、Mask 文字列は1つ以上のマスク要素で構成される必要があります。コントロールに表示される他の要素として、リテラルとプロンプトがあります。TextMaskFormat プロパティで指定されている場合は、これらの要素も使用できます。

次の表は、いくつかのサンプルマスクを示します。

マスク	動作
00/00/0000	国際的な日付書式の日付（日、月の数値、年）。「/」文字は論理日付区切りです。ユーザーには、アプリケーションの現在のカルチャに合わせた日付区切りが表示されます。
00->L<LL-0000	米国の書式での日付（日、月の省略名、年）。この書式では、月名が3文字の省略形で表示され、先頭が大文字、後の2文字が小文字になります。
(999)-000-0000	米国の電話番号（市外局番はオプション）。オプションの文字を入力しない場合は、スペースを入力するか、マスク内の最初の0の位置にマウスポインタを直接置きます。
$999,999.00	0～ 999999 の範囲の通貨の値。通貨記号、桁区切り記号、小数点記号は、実行時にカルチャ固有の記号に置き換えられます。

TextMaskFormat プロパティに次の要素のいずれかを設定して、マスクに含まれる内容を定義できます。

オプション	説明
IncludePrompt	ユーザーによるテキスト入力と、すべてのプロンプト文字を返します。
IncludeLiterals	ユーザーによるテキスト入力と、マスクで定義されているすべてのリテラル文字を返します。
IncludePromptAndLiterals	ユーザーによるテキスト入力と、マスクで定義されているすべてのリテラル文字およびすべてのプロンプト文字を返します。
ExcludePromptAndLiterals	ユーザーによるテキスト入力のみを返します。

以下のトピックでは、使用または表示できるマスク要素、リテラル要素、プロンプト要素について詳しく説明します。

マスク要素

MaskedTextBox for UWP は、Microsoft によって定義される標準の数値書式文字列をサポートします。Mask 文字列は、1つ以上のマスク要素で構成される必要があります。マスク要素については、次の表で説明します。

要素	説明
0	必須の数字。この要素は単一の数字（0～9）を表します。
9	オプションの数字またはスペース。
#	オプションの数字またはスペース。マスク内のこの位置が空白の場合、Text プロパティにはスペースとしてレンダリングされます。プラス記号（+）とマイナス記号（-）を使用できます。
L	必須の英字。入力は ASCII 文字の a ～ z と A ～ Z に制限されます。このマスク要素は、正規表現の [a-zA-Z] と同じです。
?	オプションの英字。入力は ASCII 文字の a ～ z と A ～ Z に制限されます。このマスク要素は、正規表現の [a-zA-Z]? と同じです。
&	必須の文字。
C	オプションの文字。任意の非制御文字。
A	オプションの英数字。

Copyright © GrapeCity, Inc. All rights reserved.

a	オプションの英数字。
.	小数点のプレースホルダ。実際に使用される表示文字は、書式プロバイダに合わせた小数点記号になります。
,	桁区切りのプレースホルダ。実際に使用される表示文字は、書式プロバイダに合わせた桁区切りのプレースホルダになります。
:	時刻の区切り。実際に使用される表示文字は、書式プロバイダに合わせた時刻記号になります。
/	日付の区切り。実際に使用される表示文字は、書式プロバイダに合わせた日付記号になります。
$	通貨記号。実際に表示される文字は、書式プロバイダに合わせた通貨記号になります。
<	シフトダウン。後続のすべての文字を小文字に変換します。
>	シフトアップ。後続のすべての文字を大文字に変換します。
|	直前のシフトアップまたはシフトダウンを無効にします。
****	エスケープ。マスク文字をエスケープしてリテラルにします。「\\」はバックスラッシュのエスケープシーケンスです。
その他の文字	リテラル。すべての非マスク要素は、C1MaskedTextBox 内にそのまま表示されます。リテラルは実行時に常にマスク内の静的位置を占め、ユーザーはリテラルを移動したり削除することはできません。

小数点記号(.)、桁区切り記号(,)、時刻記号(:)、日付記号(/)、通貨記号($)には、デフォルトでアプリケーションのカルチャで定義された記号が表示されます。

リテラル

「マスクの書式設定」で定義されたマスク要素に加えて、他の文字をマスクに入れることができます。これらの文字をリテラルと呼びます。リテラルは、C1MaskedTextBox 内にそのまま表示される非マスク要素です。リテラルは実行時に常にマスク内の静的位置を占め、ユーザーはリテラルを移動したり削除することはできません。

たとえば、**Mask** プロパティを「(999)-000-0000」に設定して電話番号を定義する場合、マスク文字としては「9」要素と「0」要素があります。その他の文字、つまりダッシュと丸かっこはリテラルです。これらの文字は、C1MaskedTextBox コントロールにそのまま表示されます。

リテラルを使用する場合は、**TextMaskFormat** プロパティを **IncludeLiterals** または **IncludePromptAndLiterals** に設定する必要があります。リテラルを使用しない場合は、TextMaskFormat を **IncludePrompt** または **ExcludePromptAndLiterals** に設定します。

プロンプト

C1MaskedTextBox コントロールにプロンプト文字を入れるように選択できます。プロンプト文字は、ユーザーにテキストの入力を求めるためにコントロールに表示されるテキストを定義します。プロンプト文字は、ユーザーにテキストを入力できることを示したり、入力できるテキストの種類を説明するために使用できます。デフォルトでは、下線文字「_」が使用されます。

プロンプト文字を使用する場合は、**TextMaskFormat** プロパティを **IncludePrompt** または **IncludePromptAndLiterals** に設定する必要があります。プロンプト文字を使用しない場合は、TextMaskFormat を **IncludeLiterals** または **ExcludePromptAndLiterals** に設定します。

ウォーターマーク

Watermark プロパティを使用して、どのような値を **C1MaskedTextBox** コントロールに入力する必要があるかを説明する簡単なヒントをユーザーに提供できます。ウォーターマークは、テキストが入力されるまでコントロールに表示されます。ウォーターマークを追加するには、任意の **C1MaskedTextBox** コントロールの XAML マークアップで、`Watermark="Watermark Text"` を `<Xaml:C1MaskedTextBox>` タグに追加します。

Copyright © GrapeCity, Inc. All rights reserved.

たとえば、`Watermark="テキストを入力する"` を `<Xaml:C1MaskedTextBox>` タグに追加します。次のようになります。

```
<Xaml:C1MaskedTextBox Height="23" Width="120" Name="C1MaskedTextBox1" Watermark="テキストを入力する" />
```

コントロール内をクリックしてテキストを入力すると、ウォーターマークが消えることがわかります。

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 UWP によるプログラミングに精通しており、C1NumericBox および C1MaskedTextBox 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。このセクションの各トピックは、**Input 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。

また、タスク別ヘルプの各ヘルプトピックは、新しい UWP プロジェクトが既に作成されていることを前提としています。

C1NumericBox タスク別ヘルプ

このセクションの各トピックは、**C1NumericBox** コントロールを使用して特定のタスクを実行するためのソリューションを提供します。

開始値の設定

Value プロパティは、現在選択されている値を決定します。デフォルトでは、**C1NumericBox** コントロールは、最初に Value が**0**に設定されますが、この値はカスタマイズできます。

設計時

Value プロパティを設定するには、次の手順に従います。

1. C1NumericBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、Value プロパティの横にあるテキストボックスに値（たとえば、123）を入力します。
   これで、Value プロパティは指定された値に設定されます。

XAML の場合

たとえば、Value プロパティを設定するには、次のように示すように `Value="123"` を `<Xaml:C1NumericBox>` タグに追加します。次のようになります。

マークアップ

```
<Xaml:C1NumericBox x:Name="C1NumericBox1" Value="123"></Xaml:C1NumericBox>
```

コードの場合

たとえば、Value プロパティを設定するには、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

```
C1NumericBox1.Value = 123
```

C# コードの書き方

Copyright © GrapeCity, Inc. All rights reserved.

C#

```
c1NumericBox1.Value = 123;
```

プロジェクトの実行と確認

最初に **123**（または選択した数値）がコントロールに表示されます。

インクリメント値の設定

Increment プロパティは、実行時に**アップ**ボタンまたは**ダウン**ボタンを1回押したときの **Value** プロパティの変化量を決定します。デフォルトでは、C1NumericBox コントロールは、最初に Increment が1に設定されますが、この値はカスタマイズできます。

設計時

Increment プロパティを設定するには、次の手順に従います。

1. C1NumericBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、Increment プロパティの横にあるテキストボックスに値（たとえば、20）を入力します。
   これで、Increment プロパティは指定された値に設定されます。

XAML の場合

たとえば、Increment プロパティを 20 に設定するには、`<Xaml:C1NumericBox>` タグに `Increment="20"` を追加します。次のようになります。

マークアップ

```
<Xaml:C1NumericBox x:Name="C1NumericBox1" Increment="20"> </Xaml:C1NumericBox>
```

コードの場合

たとえば、Increment プロパティを **20** 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

```
C1NumericBox1.Increment = 20
```

C# コードの書き方

C#

```
c1NumericBox1.Increment = 20;
```

プロジェクトの実行と確認

アップボタンと**ダウン**ボタンをそれぞれ数回押します。Value が 20 ずつ変化します。ただし、テキストボックスを選択し、その間の値を入力することで、直接値を編集することもできます。

最小値および最大値の設定

Minimum プロパティと **Maximum** プロパティを使用して、実行時の制限になる数値範囲を設定できます。Minimum プロパ

Copyright © GrapeCity, Inc. All rights reserved.

ティや Maximum プロパティを設定すると、ユーザーは Minimum より大きい値や Maximum より小さい値を選択できなくなります。

> **メモ:** Minimum および Maximum プロパティを設定する場合は、Minimum を Maximum より小さくする必要があります。また、Value プロパティには、Minimum と Maximum の範囲内の値を設定してください。次の例では、デフォルト値 0が、選択した範囲に入っています。

設計時

Minimum と Maximum を設定するには、次の手順に従います。

1. C1NumericBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、Maximum プロパティの横に値(たとえば、**500**)を入力します。
3. [プロパティ]タブで、Minimum の横に値(たとえば、**-500**)を入力します。
   これで、Minimum と Maximum の値が設定されます。

XAML の場合

XAML で Minimum と Maximum を設定するには、`Maximum="500" Minimum="-500"` を `<Xaml:C1NumericBox>` タグに追加します。次のようになります。

マークアップ
<Xaml:C1NumericBox x:Name="C1NumericBox1" Maximum="500" Minimum="-500"> </Xaml:C1NumericBox>

コードの場合

Minimum と Maximum を設定するには、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
C1NumericBox1.Minimum = -500 C1NumericBox1.Maximum = 500

C# コードの書き方

C#
c1NumericBox1.Minimum = -500; c1NumericBox1.Maximum = 500;

プロジェクトの実行と確認

実行時に、選択された範囲に値が制限されます。

アップ/ダウンボタンの非表示

デフォルトでは、ユーザーがボックス内の値をインクリメントおよびデクリメントできるように、**C1NumericBox** コントロールにはボタンが表示されます。C1NumericBox コントロールの**アップ**ボタンと**ダウン**ボタンを実行時に非表示にすることもできます。**アップ**ボタンと**ダウン**ボタンを非表示にするには、**ShowButtons** プロパティを **False** に設定します。

設計時

アップボタンと**ダウン**ボタンを非表示にするには、次の手順に従います。

Copyright © GrapeCity, Inc. All rights reserved.

1. C1NumericBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、ShowButtons チェックボックスをオフにします。
   これで、ShowButtons プロパティが **False** に設定されます。

XAML の場合

たとえば、XAML で**アップ**ボタンと**ダウン**ボタンを非表示にするには、`ShowButtons="False"` を `<Xaml:C1NumericBox>` タグに追加します。次のようになります。

マークアップ
<Xaml:C1NumericBox x:Name="C1NumericBox1" Width="40" Height="25" ShowButtons="False"> </Xaml:C1NumericBox>

コードの場合

たとえば、**アップ**ボタンと**ダウン**ボタンを非表示にするには、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
C1NumericBox1.ShowButtons = False

C# コードの書き方

C#
c1NumericBox1.ShowButtons = false;

プロジェクトの実行と確認

アップボタンと**ダウン**ボタンは非表示になります。

コントロールの編集のロック

デフォルトでは、**C1NumericBox** コントロールの **Value** プロパティは、実行時にユーザーによって編集可能です。コントロールの編集をロックするには、**IsReadOnly** プロパティを **True** に設定します。

設計時

C1NumericBox コントロールの実行時の編集をロックするには、次の手順に従います。

1. C1NumericBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、IsReadOnly チェックボックスをオンにします。
   これで、IsReadOnly プロパティが **False** に設定されます。

XAML の場合

たとえば、XAML で**アップ**ボタンと**ダウン**ボタンを非表示にするには、`IsReadOnly="True"` を `<Xaml:C1NumericBox>` タグに追加します。次のようになります。

マークアップ
<Xaml:C1NumericBox x:Name="C1NumericBox1" IsReadOnly="True"> </Xaml:C1NumericBox>

コードの場合

Copyright © GrapeCity, Inc. All rights reserved.

たとえば、**アップ**ボタンと**ダウン**ボタンを非表示にするには、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
C1NumericBox1.IsReadOnly = True
```

C# コードの書き方

```
C#
c1NumericBox1.IsReadOnly = true;
```

プロジェクトの実行と確認

コントロールの編集がロックされます。**アップ**ボタンと**ダウン**ボタンは淡色表示され、非アクティブになります。

C1MaskedTextBox タスク別ヘルプ

このセクションの各トピックは、**C1MaskedTextBox** コントロールを使用して特定のタスクを実行するためのソリューションを提供します。

値の設定

Value プロパティは、現在表示されているテキストを決定します。デフォルトでは、**C1MaskedTextBox** コントロールに Value が設定されていませんが、この値をカスタマイズできます。

設計時

Value プロパティを設定するには、次の手順に従います。

1. C1MaskedTextBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、Value プロパティの横にあるテキストボックスに値（たとえば、123）を入力します。
   これで、Value プロパティは指定された値に設定されます。

XAML の場合

Value プロパティを設定するには、`Value="123"` を `<Xaml:C1MaskedTextBox>` タグに追加します。次のようになります。

```
マークアップ
<Xaml:C1MaskedTextBox Height="23" Width="120" Name="C1MaskedTextBox1" Value="123">
</Xaml:C1MaskedTextBox>
```

コードの場合

Value プロパティを設定するには、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
C1MaskedTextBox1.Value = "123"
```


Copyright © GrapeCity, Inc. All rights reserved.

C# コードの書き方

```
C#
c1MaskedTextBox1.Value = "123";
```

プロジェクトの実行と確認

最初に **123**（または選択した数値）がコントロールに表示されます。

通貨のマスクの追加

Mask プロパティを使用して、通貨値のマスクを簡単に追加できます。デフォルトでは、C1MaskedTextBox コントロールに Mask が設定されていませんが、この値をカスタマイズできます。マスク文字の詳細については、「マスク要素」を参照してください。

設計時

Mask プロパティを設定するには、次の手順に従います。

1. C1MaskedTextBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、Mask プロパティの横にあるテキストボックスに「$999,999.00」と入力します。
   これで、Mask プロパティは指定された値に設定されます。

XAML の場合

Mask プロパティを設定するには、`Mask="$999,999.00"` を `<Xaml:C1MaskedTextBox>` タグに追加します。次のようになります。

```
マークアップ
<Xaml:C1MaskedTextBox Height="23" Width="120" Name="C1MaskedTextBox1" Mask="$999,999.00">
</Xaml:C1MaskedTextBox>
```

コードの場合

Value プロパティを設定するには、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

```
マークアップ
C1MaskedTextBox1.Mask = "$999,999.00"
```

C# コードの書き方

```
マークアップ
c1MaskedTextBox1.Mask = "$999,999.00";
```

プロジェクトの実行と確認

コントロールにマスクが表示されます。数値を入力します。マスクが数字に置き換わることがわかります。

プロンプト文字の変更

Copyright © GrapeCity, Inc. All rights reserved.

PromptChar プロパティは、C1MaskedTextBox コントロールでユーザーに入力を求めるために使用される文字を設定します。デフォルトでは、PromptChar プロパティは下線文字(「_」)に設定されますが、これはカスタマイズできます。PromptChar プロパティの詳細については、「プロンプト」を参照してください。

設計時

PromptChar プロパティを設定するには、次の手順に従います。

1. C1MaskedTextBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、Mask プロパティの横にあるテキストボックスに「0000」と入力して、マスクを設定します。
3. プロパティウィンドウで、PromptChar プロパティの横にあるテキストボックスに「#」(シャープ文字)を入力します。

XAML の場合

PromptChar プロパティを設定するには、`Mask="0000" PromptChar="#"` を `<Xaml:C1MaskedTextBox>`タグに追加します。次のようになります。

マークアップ

```
<Xaml:C1MaskedTextBox Height="23" Name="C1MaskedTextBox1" Width="120" Mask="0000" PromptChar="#">
</Xaml:C1MaskedTextBox>
```

コードの場合

PromptChar プロパティを設定するには、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

```
Dim x As Char = "#"c
C1MaskedTextBox1.Mask = "0000"
C1MaskedTextBox1.PromptChar = x
```

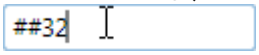
C# コードの書き方

C#

```
char x = '#';
this.c1MaskedTextBox1.Mask = "0000";
this.c1MaskedTextBox1.PromptChar = x;
```

プロジェクトの実行と確認

コントロールには、プロンプトとしてシャープ文字が表示されます。次の図では、数値 32 がコントロールに入力されています。



フォントタイプおよびフォントサイズの変更

テキストのプロパティを使用して、グリッド内のテキストの外観を変更できます。

設計時

コントロールのフォントを Arial の 10 ポイントに変更するには、次の手順に従います。

1. C1MaskedTextBox コントロールをクリックして選択します。

Copyright © GrapeCity, Inc. All rights reserved.

2. [プロパティ]タブに移動し、**FontFamily** プロパティを「Arial」(または選択したフォント)に設定します。
3. プロパティウィンドウで、**FontSize** プロパティを **10** に設定します。
   これで、コントロールのフォントサイズとフォントスタイルが設定されます。

XAML の場合

たとえば、XAML でコントロールのフォントを 10 ポイントの Arial に変更するには、`<Xaml:C1MaskedTextBox>` タグに `FontFamily="Arial" FontSize="10"` を追加します。次のようになります。

マークアップ

```
<Xaml:C1MaskedTextBox Height="23" Name="C1MaskedTextBox1" Width="120" FontSize="10"
FontFamily="Arial"></Xaml:C1MaskedTextBox>
```

コードの場合

たとえば、コントロールのフォントを 10 ポイントの Arial に変更するには、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

```
C1MaskedTextBox1.FontSize = 10
C1MaskedTextBox1.FontFamily = New System.Windows.Media.FontFamily("Arial")
```

C# コードの書き方

C#

```
c1MaskedTextBox1.FontSize = 10;
c1MaskedTextBox1.FontFamily = new System.Windows.Media.FontFamily("Arial");
```

プロジェクトの実行と確認

コントロールのコンテンツが Arial の 10 ポイントのフォントで表示されます

コントロールの編集のロック

デフォルトでは、**C1MaskedTextBox** コントロールの **Value** プロパティは、実行時にユーザーによって編集可能です。コントロールの編集をロックするには、**IsReadOnly** プロパティを **True** に設定します。

設計時

C1MaskedTextBox コントロールの実行時の編集をロックするには、次の手順に従います。

1. C1MaskedTextBox コントロールをクリックして選択します。
2. [プロパティ]タブに移動し、**IsReadOnly** チェックボックスをオンにします。
   これで、**IsReadOnly** プロパティが **False** に設定されます。

XAML の場合

XAML で C1MaskedTextBox コントロールの実行時の編集をロックするには、`IsReadOnly="True"` を `<Xaml:C1MaskedTextBox>` タグに追加します。次のようになります。

マークアップ

```
<Xaml:C1MaskedTextBox Height="23" Name="C1MaskedTextBox1" Width="120" IsReadOnly="True">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
</Xaml:C1MaskedTextBox>
```

コードの場合

C1MaskedTextBox コントロールの実行時の編集をロックするには、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
C1MaskedTextBox1.IsReadOnly = True
```

C# コードの書き方

```
C#
c1MaskedTextBox1.IsReadOnly = true;
```

プロジェクトの実行と確認

コントロールの編集はロックされます。コントロール内でカーソルをクリックしてみてください。コントロールにテキストの挿入ポイント（点滅する垂直線）が表示されないことがわかります。

Menu for UWP

Menu for UWP を使用して、タッチ操作向きのコンテキストメニューと従来の［ファイル］メニューシステムを Windows ストアアプリケーションに追加できます。**C1Menu** コントロールと **C1ContextMenu** コントロールは、深くネストされた項目と垂直方向をサポートする従来どおりの外観のメニューを備えた、リアルなデスクトップのようなルックアンドフィールを提供します。

Menu for UWP には、次のコントロールがあります。

- **C1ContextMenu**
  C1ContextMenu は、選択されたオブジェクトに関連付けられてよく使用されるコマンドを提供するポップアップメニューを提供します。
- **C1Menu**
  C1Menu は、イベントハンドラに関連付けられた要素の階層構造を表すコントロールです。

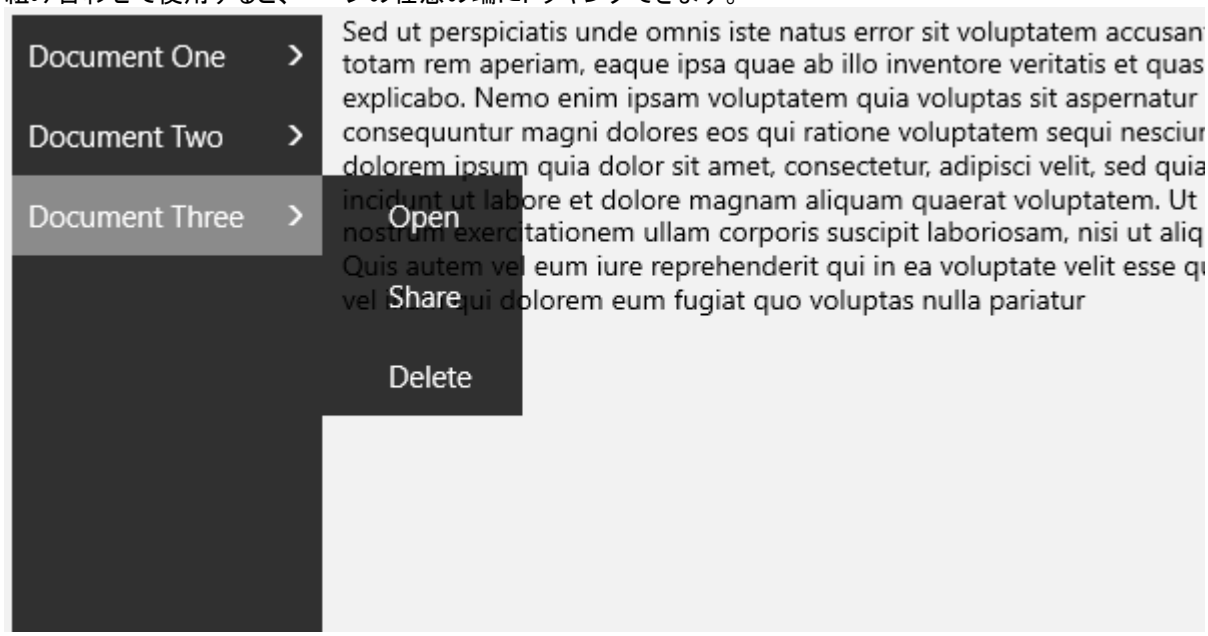
主な特長

Menu for UWP では、**C1Menu** コントロールと **C1ContextMenu** コントロールを使用して、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。**Menu for UWP** は、次の主な特長を備えています。

- **押下インジケータの表示**
  C1ContextMenu コントロールでは、押下インジケータを使用して、ユーザーが親コントロールを押したことを示すことができます。これは、Windows デスクトップのコンテキストメニューの操作感をタッチデバイス上で再現します。
- **ページ境界の検出**
  ドロップダウンメニューが自動的に、常にページ境界内に収まるように配置されます。長いメニューにはスクロールボタンが表示され、表示範囲外にメニュー項目があることを示します。
- **アイコンおよびカスタムコンテンツ**
  メニュー項目ごとにアイコンや任意のカスタムコンテンツを表示できます。

Copyright © GrapeCity, Inc. All rights reserved.

- **水平方向または垂直方向**
  Orientation は **Horizontal** または **Vertical** に設定できます。C1Menu コントロールを C1DockPanel コントロールと組み合わせて使用すると、ページの任意の端にドッキングできます。



- **チェックボックス付きの項目**
  C1MenuItem にチェックボックスを付けて、機能のオン/オフ状態を示すことができます。
- **ClearStyle を使用して簡単に色を変更する**
  C1Menu コントロールは、テンプレートを上書きしなくてもコントロールのブラシを簡単に変更できる ClearStyle をサポートします。Visual Studio でブラシのプロパティをいくつか設定するだけで、コントロールの各部のスタイルを簡単に設定できます。

クイックスタート

Copyright © GrapeCity, Inc. All rights reserved.

このクイックスタートガイドは、**Menu 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、**>C1Menu** コントロールと **>C1ContextMenu** コントロールを含む新しいプロジェクトを作成します。また、コントロールにメニュー項目、サブメニュー、コンテキストメニューを追加します。

手順1：Windows ストアアプリケーションの作成

この手順では、**>C1Menu** コントロールと **>C1ContextMenu** コントロールを使用して、Windows ストアアプリケーションを作成します。

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、言語の下で[**Windows ストア**]を選択し、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. ソリューションエクスプローラでプロジェクト名を右クリックし、[**参照の追加**]を選択します。
4. [**参照マネージャ**]ダイアログボックスで[**ComponentOne for UWP**]を選択します。[**OK**]をクリックしてダイアログボックスを閉じ、参照を追加します。
5. **MainPage.xaml.cs**（または **MainPage.xaml.vb**）コードファイルを開き、ページの先頭に次の参照を追加します。

   Visual Basic コードの書き方

   Visual Basic
   ```
   Imports C1.Xaml
   ```

   C# コードの書き方

   C#
   ```
   using C1.Xaml;
   ```
6. **MainPage.xaml** を開き、`<Page>` タグ内に次のマークアップを追加します。

   マークアップ
   ```
   xmlns:C1="using:C1.Xaml"
   ```
   これにより、C1.Xaml アセンブリへの参照がプロジェクトに追加されます。
7. `<Page>` タグと `</Page>` タグの間に次のマークアップを追加します。

   マークアップ
   ```
   <Page.Resources>
       <Style TargetType="Image" x:Key="MenuIcon">
           <Setter Property="Width" Value="16"/>
           <Setter Property="Height" Value="16"/>
           <Setter Property="Margin" Value="5 0 0 0"/>
       </Style>
       <Style TargetType="TextBlock" x:Key="TextIconStyle">
           <Setter Property="FontSize" Value="20" />
           <Setter Property="FontFamily" Value="Segoe UI Symbol" />
           <Setter Property="FontWeight" Value="Normal" />
           <Setter Property="Foreground" Value="{StaticResource AppBarItemForegroundThemeBrush}" />
           <Setter Property="HorizontalAlignment" Value="Center" />
           <Setter Property="VerticalAlignment" Value="Center" />
           <Setter Property="Margin" Value="5,-1,0,0"/>
       </Style>
       <Style TargetType="C1:C1MenuItem">
   ```


Copyright © GrapeCity, Inc. All rights reserved.

```
        <Setter Property="ItemContainerTransitions">
          <Setter.Value>
            <TransitionCollection>
              <RepositionThemeTransition/>
              <EntranceThemeTransition/>
            </TransitionCollection>
          </Setter.Value>
        </Setter>
        <Setter Property="ItemContainerTransitions">
          <Setter.Value>
            <TransitionCollection>
              <RepositionThemeTransition/>
              <EntranceThemeTransition/>
            </TransitionCollection>
          </Setter.Value>
        </Setter>
      </Style>
    </Page.Resources>
```

このマークアップは、スタイルリソースを追加します。

8. `<Grid>` タグと `</Grid>` タグの間に次のマークアップを追加します。

マークアップ

```
<Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition />
    </Grid.RowDefinitions>
```

このマークアップは、行定義をグリッドに追加します。

これで、**Menu for UWP** クイックスタートの最初の手順は終了です。この手順では、Windows ストアプロジェクトを作成しました。次の手順では、コントロールにタブとタブページを追加します。

手順2：アプリケーションへの C1Menu の追加

前の手順では、Windows ストアアプリケーションを作成しました。この手順では、**C1Menu** コントロールを追加します。

1. **MainPage.xaml** で、`<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。
2. C1DockPanel コントロールのマークアップを追加します。

マークアップ

```
<C1:C1DockPanel Grid.Row="1" LastChildFill="False"></C1:C1DockPanel>
```

このコントロール内に **C1Menu** を追加します。

3. `</C1:C1DockPanel>` タグの直後に **TextBlock** のマークアップを追加します。

マークアップ

```
<TextBlock x:Name="txt" Foreground="Red" Text="" FontSize="16" VerticalAlignment="Bottom"
HorizontalAlignment="Center" Margin="10" />
```

実行時には、選択したメニュー項目の名前がこの **TextBlock** に表示されます。

4. `<C1:C1DockPanel>` タグ内に次のマークアップを追加します。

マークアップ

```
<C1:C1Menu x:Name="VisualStudioMenu" C1:C1DockPanel.Dock="Top" DetectBoundaries="True"
MinWidth="200" ItemClick="Menu_ItemClick">
  <C1:C1Menu.ItemContainerTransitions>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        <TransitionCollection>
            <EntranceThemeTransition/>
        </TransitionCollection>
    </C1:C1Menu.ItemContainerTransitions>
</C1:C1Menu>
```

これにより、C1Menu コントロールが追加されます。

5. </c1:C1Menu> タグの直前に次のマークアップを追加します。

マークアップ

```
</c1:C1Menu> タグの直前に次のマークアップを追加します。
<C1:C1MenuItem Header="ファイル">
<C1:C1MenuItem Header="新規作成">
    <C1:C1MenuItem Header="プロジェクト..." IsCheckable="True" IsChecked="True" >
        <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE188;" />
        </C1:C1MenuItem.Icon>
    </C1:C1MenuItem>
    <C1:C1MenuItem Header="ウェブサイト..." >
        <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE12B;" />
        </C1:C1MenuItem.Icon>
    </C1:C1MenuItem>
    <C1:C1MenuItem Header="チームプロジェクト..." >
        <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE125;" />
        </C1:C1MenuItem.Icon>
    </C1:C1MenuItem>
    <C1:C1MenuItem Header="ファイル..." >
        <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE132;" />
        </C1:C1MenuItem.Icon>
    </C1:C1MenuItem>
</C1:C1MenuItem>
<C1:C1MenuItem Header="開く">
    <C1:C1MenuItem Header="プロジェクト..." >
        <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE19C;" />
        </C1:C1MenuItem.Icon>
    </C1:C1MenuItem>
    <C1:C1MenuItem Header="ウェブサイト..." >
        <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE12B;" />
        </C1:C1MenuItem.Icon>
    </C1:C1MenuItem>
    <C1:C1MenuItem Header="ファイル..." >
        <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE19C;" />
        </C1:C1MenuItem.Icon>
    </C1:C1MenuItem>
</C1:C1MenuItem>
<C1:C1Separator />
<C1:C1MenuItem Header="閉じる" />
<C1:C1MenuItem Header="ソリューションを閉じる">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE10A;" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
<C1:C1Separator />
<C1:C1MenuItem Header="保存" >
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE105;" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
<C1:C1MenuItem Header="すべてを保存" />
<C1:C1Separator />
<C1:C1MenuItem Header="ページセットアップ">
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE160;" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
<C1:C1MenuItem Header="印刷">
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#x2399;" FontWeight="ExtraBold" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
<C1:C1Separator />
<C1:C1MenuItem Header="終了"/>
</C1:C1MenuItem>
<C1:C1MenuItem Header="編集">
<C1:C1MenuItem Header="元に戻す">
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE10E;" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
<C1:C1MenuItem Header="やり直し">
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE10D;" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
<C1:C1Separator />
<C1:C1MenuItem Header="切り取り">
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE16B;" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
<C1:C1MenuItem Header="コピー">
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE16F;" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
<C1:C1MenuItem Header="貼り付け">
    <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE16D;" />
    </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
<C1:C1MenuItem Header="削除" Width="100">
   <C1:C1MenuItem.Icon>
      <TextBlock Text="&#xE106;" />
   </C1:C1MenuItem.Icon>
</C1:C1MenuItem>
</C1:C1MenuItem>
<C1:C1MenuItem Header="ウィンドウ">
<C1:C1MenuItem Header="ドキュメント 1" IsCheckable="True" IsChecked="True" />
<C1:C1MenuItem Header="ドキュメント 2" IsCheckable="True" IsChecked="True" />
<C1:C1MenuItem Header="ドキュメント 3" IsCheckable="True" />
<C1:C1MenuItem Header="ドキュメント 4" IsCheckable="True" />
<C1:C1Separator />
<C1:C1MenuItem Header="排他的なチェック 1" GroupName="Exclusives" IsCheckable="True"
IsChecked="True" />
<C1:C1MenuItem Header="排他的なチェック 2" GroupName="Exclusives" IsCheckable="True"
IsChecked="True" />
<C1:C1MenuItem Header="排他的なチェック 3" GroupName="Exclusives" IsCheckable="True" />
</C1:C1MenuItem>
<C1:C1MenuItem Header="アイテム">
<C1:C1MenuItem Header="サブアイテム 1">
   <C1:C1MenuItem Header="サブアイテム 2">
      <C1:C1MenuItem Header="サブアイテム 3">
         <C1:C1MenuItem Header="サブアイテム 4">
            <C1:C1MenuItem Header="サブアイテム 5">
            </C1:C1MenuItem>
         </C1:C1MenuItem>
      </C1:C1MenuItem>
   </C1:C1MenuItem>
</C1:C1MenuItem>
</C1:C1MenuItem>
```

上のマークアップは、メニュー構造のマークアップを追加します。

6. **MainPage.xaml.cs**（または **MainPage.xaml.vb**）ページを開き、プロジェクトに次の **Click** イベントハンドラを追加します。

Visual Basic コードの書き方

Visual Basic

```
Private Sub Menu_ItemClick(sender As Object, e As C1.Xaml.SourcedEventArgs)
  txt.Text = "クリックされたアイテム: " & DirectCast(e.Source, C1.Xaml.C1MenuItem).Header.ToString()
End Sub
```

C# コードの書き方

C#

```
private void Menu_ItemClick(object sender, C1.Xaml.SourcedEventArgs e)
      {
         txt.Text = "クリックされたアイテム: " + ((C1.Xaml.C1MenuItem)e.Source).Header.ToString();
      }
```

この手順では、C1Menu コントロールを追加しました。次の手順では、C1ContextMenu コントロールをアプリケーションに追加します。

Copyright © GrapeCity, Inc. All rights reserved.

手順3：C1Menu コントロールへのC1ContextMenu の追加

前の手順では、**C1Menu** コントロールのメニュー項目にサブメニューを追加しました。この手順では、この C1Menu コントロールに1つの **C1ContextMenu** コントロールを追加します。このコンテキストメニューには項目が1つあります。これをクリックすると、「手順2：アプリケーションへの C1Menu の追加」で作成した **C1Menu** コントロールの最上位の［**Added Items**］メニュー項目に、いくつかのサブメニュー項目が追加されます。

次の手順に従います。

1. XAML ビューで、`</C1:C1Menu>` タグの直前に次の XAML マークアップを配置します。

マークアップ

```
<C1:C1ContextMenuService.ContextMenu>
   <C1:C1ContextMenu x:Name="contextMenu" ItemClick="Menu_ItemClick">
      <C1:C1ContextMenu.ItemContainerTransitions>
         <TransitionCollection>
            <EntranceThemeTransition FromVerticalOffset="10" FromHorizontalOffset="0"
IsStaggeringEnabled="False"/>
         </TransitionCollection>
      </C1:C1ContextMenu.ItemContainerTransitions>
      <C1:C1MenuItem Header="追加">
         <C1:C1MenuItem.ItemContainerTransitions>
            <TransitionCollection>
               <EntranceThemeTransition FromVerticalOffset="10" FromHorizontalOffset="0"
IsStaggeringEnabled="False"/>
            </TransitionCollection>
         </C1:C1MenuItem.ItemContainerTransitions>
         <C1:C1MenuItem Header="新しい項目">
            <C1:C1MenuItem.Icon>
               <TextBlock Text="&#xE1DA;" />
            </C1:C1MenuItem.Icon>
         </C1:C1MenuItem>
         <C1:C1MenuItem Header="終了"/>
         <C1:C1MenuItem Header="フォルダ"/>
         <C1:C1Separator />
         <C1:C1MenuItem Header="Class"/>
      </C1:C1MenuItem>
      <C1:C1Separator />
      <C1:C1MenuItem Header="プロジェクトから除外"/>
      <C1:C1Separator />
      <C1:C1MenuItem Header="切り取り">
         <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE16B;" />
         </C1:C1MenuItem.Icon>
      </C1:C1MenuItem>
      <C1:C1MenuItem Header="コピー">
         <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE16F;" />
         </C1:C1MenuItem.Icon>
      </C1:C1MenuItem>
      <C1:C1MenuItem Header="貼り付け">
         <C1:C1MenuItem.Icon>
            <TextBlock Text="&#xE16D;" />
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            </C1:C1MenuItem.Icon>
        </C1:C1MenuItem>
        <C1:C1MenuItem Header="削除">
            <C1:C1MenuItem.Icon>
                <TextBlock Text="&#xE107;" />
            </C1:C1MenuItem.Icon>
        </C1:C1MenuItem>
        <C1:C1MenuItem Header="名前の変更"/>
        <C1:C1Separator/>
        <C1:C1MenuItem Header="プロパティ"/>
    </C1:C1ContextMenu>
</C1:C1ContextMenuService.ContextMenu>
```

上のマークアップは、**C1ContextMenuService** ヘルパークラスを使用して、C1Menu コントロールに C1ContextMenu コントロールを追加します。この **C1ContextMenu** コントロールには、"Menu_ItemClick" という名前の **Click** イベントにアタッチされた C1MenuItem が1つ含まれています。

2. `<c1:C1MenuItem Header="既存の項目"/>` タグに `x:Name="AddedItems"` を追加します。これにより、コードから呼び出せる一意の識別子を項目に渡すことができます。

この手順では、C1Menu コントロールに C1ContextMenu コントロールを追加します。次の手順では、プロジェクトを実行して **Menu for UWP** クイックスタートの結果を確認します。

手順4：プロジェクトの実行

C1Menu コントロールを持つ Windows ストアプロジェクトを作成したので、この後はプロジェクトを実行して作業の結果を確認するだけです。

次の手順に従います。

1. [**デバッグ**]→[**デバッグ開始**]を選択して、プロジェクトを実行します。
2. [**ファイル**]をクリックして、サブメニューが表示されることを確認します。
3. カーソルを[**新規作成**]に置き、別のサブメニューが表示されることを確認します。
4. [**ウィンドウ**]をクリックし、[**ドキュメント 4**]をクリックします。
   [**ウィンドウ**]サブメニューが閉じ、テキストボックスに項目の名前が表示されます。
5. メニューを右クリックし、コンテキストメニューが表示されることを確認します。
6. コンテキストメニューの項目を選択し、画面の TextBlock に項目の名前が表示されることを確認します。

おめでとうございます。これで **Menu for UWP** クイックスタートは終了です。

RadialMenu for UWP

RadialMenu for UWP を使用すると、Windows ストアアプリケーションに魅力的なラジアルメニューシステムを追加できます。**C1RadialMenu** コントロールは、よく使用されているマイクロソフトアプリケーションをモデルにして、従来のコンテキストメニューに代わるユニークなタッチ操作向きのメニューを提供します。

主な特長

RadialMenu 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。**RadialMenu for UWP** は、次の主な特長を備えています。

- **メニューのネスト**
  ラジアルメニューは必要な任意の深さにネストでき、必要な数だけラジアルメニューに項目を追加できます。
  C1RadialMenu コントロールは、含まれている項目の数に基づいて、自動的にセクターを作成します。
- **柔軟な位置決め**

Copyright © GrapeCity, Inc. All rights reserved.

C1RadialMenu コントロール内の各項目の正確な位置を指定でき、項目の開始角度を指定することもできます。

- **自動選択**
  C1RadialMenu コントロールでは、各メニュー項目に任意の数のサブメニュー項目を入れることができ、各サブメニューには選択された項目が1つ表示されます。選択されたサブメニュー項目を指定することも、それまでのユーザーのアクションに基づいてコントロールに自動的に項目の選択を任せることもできます。このように、デフォルトではないがよく使用されるメニュー項目がメインメニューに表示されるため、これまでよりすばやい選択が可能です。
- **自動折りたたみ**
  デフォルトでは、ユーザーがラジアルメニューの外部をクリックしても、C1RadialMenu コントロールはサブメニューと共に開いたままです。ただし、自動折りたたみ機能を有効にして、この動作を変更できます。その場合は、コントロールの境界の外側をクリックして、ラジアルメニューを閉じることができます。
- **チェック可能なメニュー項目**
  IsCheckable を設定することで、C1RadialMenuItem をチェック可能なメニュー項目にすることができます。チェックされた項目は、通常のチェックマークではなく、強調表示された項目としてマークされます。
- **ClearStyle を使用して簡単に色を変更する**
  C1RadialMenu コントロールは、テンプレートを上書きしなくてもコントロールのブラシを簡単に変更できる ComponentOne の ClearStyle 技術をサポートします。Visual Studio でブラシのプロパティをいくつか設定するだけで、コントロールの各部のスタイルを簡単に設定できます。

クイックスタート

このクイックスタートガイドは、**RadialMenu 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、Visual Studio を使用して、**C1RadialMenu** コントロールを含む新しいプロジェクトを作成します。

手順1：C1RadialMenu アプリケーションの作成

この手順では、**C1RadialMenu** コントロールを使用して、Windows ストアアプリケーションを作成します。

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、言語の下で[**Windows ストア**]を選択し、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. ソリューションエクスプローラでプロジェクト名を右クリックし、[**参照の追加**]を選択します。
4. [**参照マネージャ**]ダイアログボックスで[**ComponentOne for UWP**]を選択します。[**OK**]をクリックしてダイアログボックスを閉じ、参照を追加します。
5. **MainPage.xaml** をまだ開いていない場合は開き、`<Page>` タグ内に次のマークアップを追加します。

   マークアップ

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

   これにより、C1.Xaml アセンブリへの参照がプロジェクトに追加されます。
6. `<Page>` タグと `</Page>` タグの間に次のマークアップを追加します。

   マークアップ

   ```
   <Page.Resources>
     <Style TargetType="TextBlock" x:Key="TextIconStyle">
       <Setter Property="Margin" Value="-10" />
       <Setter Property="FontSize" Value="20" />
       <Setter Property="FontFamily" Value="Segoe UI Symbol" />
       <Setter Property="FontWeight" Value="Normal" />
       <Setter Property="Foreground" Value="#333333" />
       <Setter Property="HorizontalAlignment" Value="Center" />
       <Setter Property="VerticalAlignment" Value="Center" />
     </Style>
   ```


Copyright © GrapeCity, Inc. All rights reserved.

```
</Page.Resources>
```

このマークアップは、メニュー項目コンテンツのレイアウトを定義するスタイルリソースを追加します。次の手順では、C1RadialMenu にこれらのメニュー項目を追加します。各メニュー項目には、画像と TextBlock ラベルが1つずつ含まれます。

7. <Grid> タグと </Grid> タグの間に次のマークアップを追加します。

マークアップ

```
<Border Background="LemonChiffon" MinHeight="40" BorderBrush="#969696" BorderThickness="1"
Padding="5" HorizontalAlignment="Stretch" VerticalAlignment="Stretch">
</Border >
```

このマークアップは、アプリケーションに境界線の定義を追加します。

これで、**RadialMenu for UWP** クイックスタートの最初の手順は終了です。この手順では、Windows ストアプロジェクトを作成しました。次の手順では、RadialMenu コントロールとメニュー項目を追加します。

手順2：コントロールへの RadialMenu 項目の追加

前の手順では、UWP プロジェクトを作成しました。この手順では、**C1RadialMenu** コントロールを追加します。

1. **MainPage.xaml** で、<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。
2. ContextMenuProperty 添付プロパティのマークアップを追加します。

マークアップ

```
<Xaml:C1ContextMenuService.ContextMenu>
</Xaml:C1ContextMenuService.ContextMenu>
```

C1ContextMenuService は、C1RadialMenu が任意のコントロールのコンテキストメニューとして動作できるようにします。コンテキストメニューは、親コントロールを右クリックまたは右タップすると自動的に表示されます。この場合、親コントロールは Grid です。次に、このコントロール内に **C1RadialMenu** を追加します。

> **メモ: C1RadialMenu** コントロールをプログラムによって表示する場合は、C1ContextMenuService を省略し、コードで単に Show メソッドを呼び出すことができます。

3. < Xaml:C1ContextMenuService.ContextMenu > タグ内に次のマークアップを追加します。

マークアップ

```
<Xaml:C1RadialMenu x:Name="contextMenu" Offset="-130,0" ItemClick="contextMenu_ItemClick"
ItemOpened="contextMenu_ItemOpened" >
 </Xaml: C1RadialMenu>
```

これにより、C1RadialMenu コントロールが追加されます。

4. <Xaml:C1RadialMenu> </Xaml:C1RadialMenu> タグ内に、次のマークアップを追加します。

マークアップ

```
<Xaml:C1RadialMenuItem Header="UndoRedo" SelectedIndex="0" ShowSelectedItem="True" Command="
{Binding UndoCommand, ElementName=page}">
    <Xaml:C1RadialMenuItem Header="元に戻す" >
      <Xaml:C1RadialMenuItem.Icon>
        <TextBlock Text="&#xE10E;" Style="{StaticResource TextIconStyle}" />
      </Xaml:C1RadialMenuItem.Icon>
    </Xaml:C1RadialMenuItem>
    <Xaml:C1RadialMenuItem Header="やり直し" >
      <Xaml:C1RadialMenuItem.Icon>
        <TextBlock Text="&#xE10D;" Style="{StaticResource TextIconStyle}" />
      </Xaml:C1RadialMenuItem.Icon>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
      </Xaml:C1RadialMenuItem>
      <Xaml:C1RadialMenuItem ToolTip="テキストの書式をクリアします" DisplayIndex="7">
      <Xaml:C1RadialMenuItem.Header>
                  <TextBlock TextWrapping="Wrap" MaxWidth="50" TextAlignment="Center">書式をクリア
</TextBlock>
               </Xaml:C1RadialMenuItem.Header>
            </Xaml:C1RadialMenuItem>
         </Xaml:C1RadialMenuItem>
         <Xaml:C1RadialMenuItem AutoSelect="True" ShowSelectedItem="True" Header="クリップボード"
SectorCount="8">
            <Xaml:C1RadialMenuItem Header="切り取り">
               <Xaml:C1RadialMenuItem.Icon>
                  <TextBlock Text="&#xE16B;" Style="{StaticResource TextIconStyle}" />
               </Xaml:C1RadialMenuItem.Icon>
            </Xaml:C1RadialMenuItem>
            <Xaml:C1RadialMenuItem Header="コピー">
               <Xaml:C1RadialMenuItem.Icon>
                  <TextBlock Text="&#xE16F;" Style="{StaticResource TextIconStyle}" />
               </Xaml:C1RadialMenuItem.Icon>
            </Xaml:C1RadialMenuItem>
            <Xaml:C1RadialMenuItem Header="貼り付け">
               <Xaml:C1RadialMenuItem.Icon>
                  <TextBlock Text="&#xE16D;" Style="{StaticResource TextIconStyle}" />
               </Xaml:C1RadialMenuItem.Icon>
            </Xaml:C1RadialMenuItem>
         </Xaml:C1RadialMenuItem>
         <Xaml:C1RadialMenuItem Header="追加" SectorCount="6">
            <Xaml:C1RadialMenuItem.Icon>
               <TextBlock Text="&#xE109;" Style="{StaticResource TextIconStyle}" />
            </Xaml:C1RadialMenuItem.Icon>
            <Xaml:C1RadialMenuItem Header="新しい項目" ToolTip="新しい項目を追加します">
               <Xaml:C1RadialMenuItem.Icon>
                  <TextBlock Text="&#xE1DA;" Style="{StaticResource TextIconStyle}" />
               </Xaml:C1RadialMenuItem.Icon>
            </Xaml:C1RadialMenuItem>
            <Xaml:C1RadialMenuItem Header="既存の項目" ToolTip="既存の項目を追加します">
               <Xaml:C1RadialMenuItem.Icon>
                  <TextBlock Text="&#xE132;" Style="{StaticResource TextIconStyle}" />
               </Xaml:C1RadialMenuItem.Icon>
            </Xaml:C1RadialMenuItem>
            <Xaml:C1RadialMenuItem Header="フォルダー">
               <Xaml:C1RadialMenuItem.Icon>
                  <TextBlock Text="&#xE188;" Style="{StaticResource TextIconStyle}" />
               </Xaml:C1RadialMenuItem.Icon>
            </Xaml:C1RadialMenuItem>
            <Xaml:C1RadialMenuItem Header="クラス">
               <Xaml:C1RadialMenuItem.Icon>
                  <TextBlock Text="&#xE1D3;" Style="{StaticResource TextIconStyle}" />
               </Xaml:C1RadialMenuItem.Icon>
            </Xaml:C1RadialMenuItem>
         </Xaml:C1RadialMenuItem>
         <Xaml:C1RadialMenuItem Header="除外" ToolTip="Exclude From Project" >
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            <Xaml:C1RadialMenuItem.Icon>
                <TextBlock Text="&#xE10A;" Style="{StaticResource TextIconStyle}" />
            </Xaml:C1RadialMenuItem.Icon>
        </Xaml:C1RadialMenuItem>
        <Xaml:C1RadialMenuItem Header="削除">
            <Xaml:C1RadialMenuItem.Icon>
                <TextBlock Text="&#xE107;" Style="{StaticResource TextIconStyle}" />
            </Xaml:C1RadialMenuItem.Icon>
        </Xaml:C1RadialMenuItem>
        <Xaml:C1RadialMenuItem Header="名前の変更">
            <Xaml:C1RadialMenuItem.Icon>
                <TextBlock Text="&#xE13E;" Style="{StaticResource TextIconStyle}" />
            </Xaml:C1RadialMenuItem.Icon>
        </Xaml:C1RadialMenuItem>
        <Xaml:C1RadialMenuItem Header="プロパティ">
            <Xaml:C1RadialMenuItem.Icon>
                <TextBlock Text="&#xE115;" Style="{StaticResource TextIconStyle}" />
            </Xaml:C1RadialMenuItem.Icon>
        </Xaml:C1RadialMenuItem>
```

上のマークアップは、RadialMenu にメニュー項目を追加します。

5. `</Xaml:C1ContextMenuService.ContextMenu>` タグのすぐ下に次のマークアップを追加します。

マークアップ

```
<TextBlock Text="ここはコンテクストメニューのボタンを押してください (Windowsの場合右クリックする)。"
Foreground="Sienna" FontSize="16" TextWrapping="Wrap" HorizontalAlignment="Center"
VerticalAlignment="Center"/>
</Border>
<TextBlock x:Name="txt" Foreground="Red" Text="" FontSize="16" VerticalAlignment="Bottom"
HorizontalAlignment="Center" Margin="10" />
```

これは、`</Border>` タグの前後に1つずつ、合計2つの汎用 TextBlock コントロールをアプリケーションに追加します。最初の TextBlock には、C1RadialMenu の表示方法についての指示が入ります。2番目の TextBlock には、ユーザーがタップ、クリック、または開いた C1RadialMenuItem が表示されます。

6. **MainPage.xaml.cs** ページを開き、プロジェクトに次の **Click** イベントハンドラを追加します。

C# コードの書き方

C#

```
private void contextMenu_ItemClick(object sender, C1.Xaml.SourcedEventArgs e)
    {
        txt.Text = "選択されました: " + ((C1.Xaml.C1RadialMenuItem)e.Source).Header.ToString();
    }
    private void contextMenu_ItemOpened(object sender, C1.Xaml.SourcedEventArgs e)
    {
        txt.Text = "開かれました: " + ((C1.Xaml.C1RadialMenuItem)e.Source).Header.ToString();
    }
```

この手順では、C1RadialMenu コントロールを追加しました。次の手順では、プロジェクトを実行して **RadialMenu for UWP** クイックスタートの結果を確認します。

Copyright © GrapeCity, Inc. All rights reserved.

手順3：プロジェクトの実行

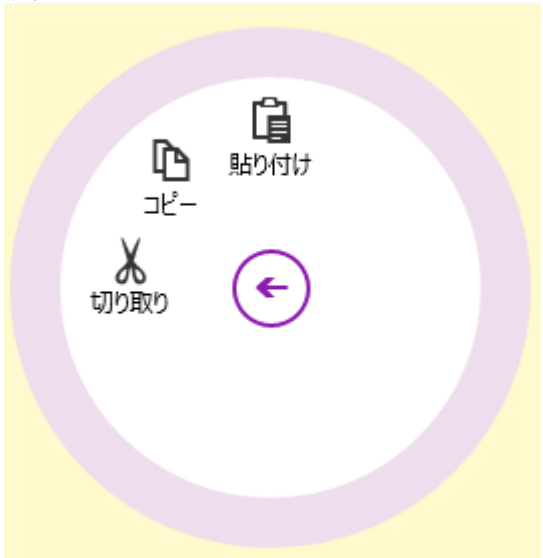
C1RadialMenu コントロールを持つ Windows ストアプロジェクトを作成したので、この後はプロジェクトを実行して作業の結果を確認するだけです。

次の手順に従います。

1. [**デバッグ**]→[**デバッグ開始**]を選択して、プロジェクトを実行します。ページを右タップまたは右クリックすると、次の図のようにナビゲーションボタンが表示されます。

2. ラジアルメニューを表示するには、ナビゲーションボタンをタップまたはクリックします。

3. [**切り取り**]C1RadialMenuItem の上の[**展開領域**]をタップし、[**クリップボード**]メニューが表示されることを確認します。



4. メインラジアルメニューに戻るには、C1RadialMenu コントロールの中心にある紫色の矢印をタップします。

おめでとうございます。これで **RadialMenu for UWP** クイックスタートは終了です。

C1RadialMenu の使い方

以下のトピックでは、**C1RadialMenu** コントロールの操作の基本について説明します。このセクションでは、コントロールの要

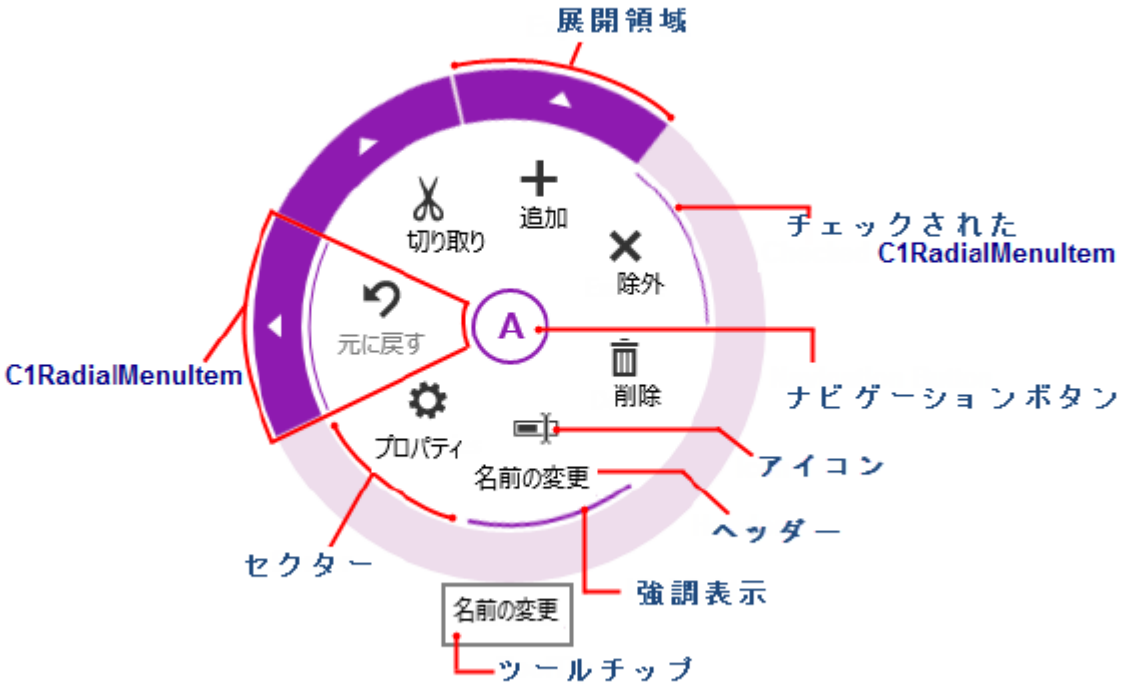
Copyright © GrapeCity, Inc. All rights reserved.

素について概説し、コントロールのよく使用される機能をいくつか説明します。

C1RadialMenu 要素

C1RadialMenu は、要素を放射状に並べることができるコントロールです。ラジアルメニューは必要な任意の深さにネストでき、必要な数だけラジアルメニューに項目を追加できます。

次の図に、C1RadialMenu コントロールの要素を示します。



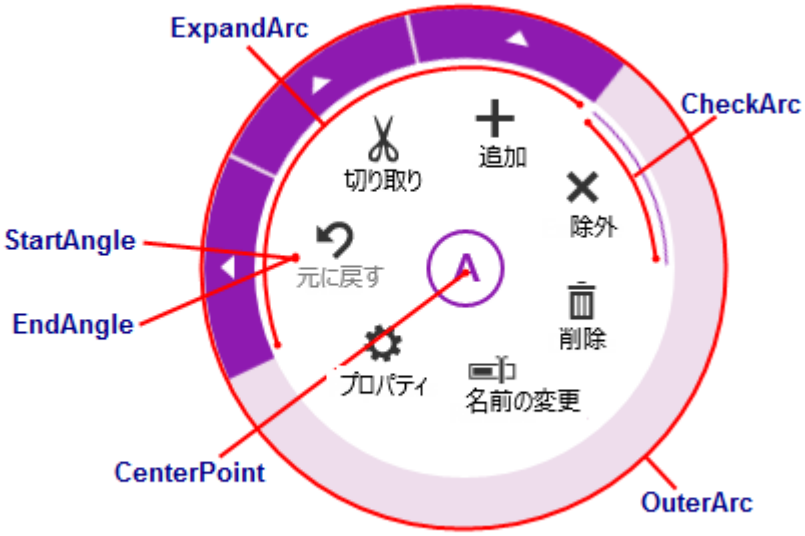
C1RadialMenu コントロールの要素には、次のものがあります。

- **C1RadialMenu**
  メインラジアルメニューは、円形の最上位のコンテキストメニューコントロールです。これは、最上位にあるいくつかの C1RadialMenuItem で構成され、C1RadialMenuItem のすべての機能を利用できます。
- **C1RadialMenuItem**
  ラジアルメニュー項目は、**C1RadialMenuItem** クラスによって表されます。C1RadialMenuItem は Header と Icon を持つことができ、プロパティを IsCheckable に設定することもできます。また、各ラジアルメニュー項目はボタンが押されたときに呼び出される Command に関連付けられている可能性があります。子項目を含むラジアルメニュー項目には、**展開領域**もあります。
- **セクター**
  C1RadialMenu は複数のセクターに分割されます。**C1RadialMenu** コントロールは、コントロールに含まれる C1RadialMenuItem の数に基づいて自動的にセクターを作成しますが、SectorCount プロパティを使用して、セクター数をカスタマイズすることもできます。
- **ナビゲーションボタン**
  ユーザーがアプリケーションを右タップすると、ナビゲーションボタンが表示されます。ナビゲーションボタンをタップして、C1RadialMenu を表示または非表示にします。ユーザーがサブメニューに移動すると、このボタンは戻る矢印に変わります。
- **アイコン**
  Icon プロパティを使用して、C1RadialMenu に表示するアイコンをカスタマイズできます。

Copyright © GrapeCity, Inc. All rights reserved.

C1RadialMenuItem 要素とC1RadialPanel 要素

C1RadialMenuItem のデフォルトコントロールテンプレートに形状パスを描画するためのプロパティがいくつかあります。デフォルトでは、**C1RadialPanel** によってこれらのプロパティが設定されますが、C1RadialPanel で明示的にプロパティを設定することで、カスタムコントロールテンプレートを作成することもできます。



C1RadialMenuItem 要素

C1RadialMenuItem 要素は、デフォルトでは C1RadialPanel によって設定されます。これらの要素は、その C1RadialMenuItem コントロール内に形状パスを描画します。デフォルトコントロールテンプレートはこれらのパスを使用し、ユーザーもカスタムコントロールテンプレートでこれらのパスを使用できます。

- **CenterPoint**
  CenterPoint は、中心の座標を取得します。これは、現在の C1RadialMenuItem を表すXAMLで円セクターを描くために使用することができます。
- **CheckArc**
  CheckArc プロパティは、チェック円弧の定義を取得します。
- **ExpandArc**
  ExpandArc プロパティは、展開領域円弧の定義を取得します。
- **OuterArc**
  OuterArc は、外側の円弧セグメントの定義を取得します。これは、現在の C1RadialMenuItem を表すXAMLで円セクターを描くために使用することができます。

C1RadialPanel 要素

C1RadialPanel は、デフォルトの **C1RadialMenu** スタイルで作成されます。これは、**C1RadialMenuItemsPresenter** によって **ItemsPanel** として使用されます。C1RadialPanel は周囲の 360 度すべてを利用して、ラジアルメニュー項目を円内に並べます。StartAngle または EndAngle を変更する必要がある場合は、次のような XAML を使用して、デフォルトのスタイルを再定義し、カスタム設定を使用することができます。

```
<Setter Property="ItemsPanel">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
 <Setter.Value>
  <ItemsPanelTemplate>
   <c1:C1RadialPanel StartAngle="-180" EndAngle="180" />
  </ItemsPanelTemplate>
 </Setter.Value>
</Setter>
```

- **StartAngle**
  StartAngle は、最初の C1RadialMenuItem が配置されるべき場所を定義します。上のサンプルの -180 は時計の文字盤の9時に対応します。これは、デフォルトの StartAngle です。
- **EndAngle**
  EndAngle は、最後の C1RadialMenuItem が配置されるべき場所を定義します。上のサンプルの 180 は時計の文字盤の9時に対応します。これは、デフォルトの EndAngle です。

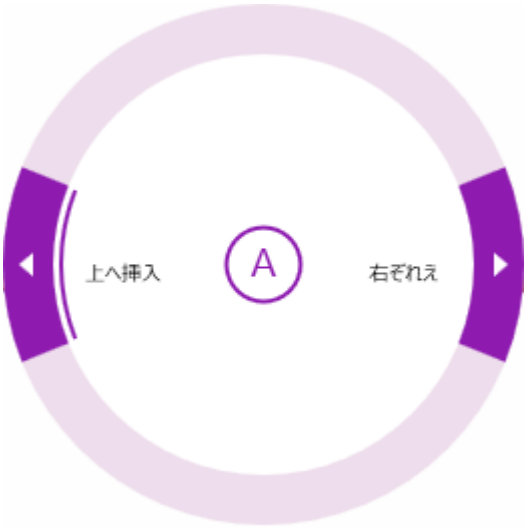
C1RadialMenu の機能

自動折りたたみ

デフォルトでは、ユーザーがラジアルメニューの外部をクリックしても、**C1RadialMenu** コントロールはサブメニューと共に開いたままです。ラジアルメニューを閉じるには、C1RadialMenu コントロールの中心にあるナビゲーションボタンをクリックする必要があります。ただし、自動折りたたみ機能を有効にして、この動作を変更できます。その場合は、コントロールの境界の外側をクリックして、ラジアルメニューを閉じることができます。自動折りたたみをオンにするには、**AutoCollapse** プロパティを **True** に設定します。

チェック可能なラジアルメニュー項目

IsCheckable を **True** に設定することで、**C1RadialMenuItem** をチェック可能な RadialMenu 項目にすることができます。チェックされた項目は、通常のチェックマークではなく、強調表示された項目としてマークされます。下の図で、チェックされた状態の[**Insert Above**]オプションを確認できます。チェックは、「C1RadialMenu 要素」で見た強調表示より線が細いことがわかります。



グループに追加する各項目の GroupName プロパティを設定することで、相互に排他的なチェック可能な項目のグループを作成できます。たとえば、下の XAML は、相互に排他的な4つのチェック可能な項目のグループを作成します。

マークアップ

```
 <Xaml:C1RadialMenu SectorCount="8" >
  <Xaml:C1RadialMenuItem Header="挿入" SectorCount="8" AutoSelect="True" ShowSelectedItem="True"
```


Copyright © GrapeCity, Inc. All rights reserved.

```
IsCheckable="True" >
    <Xaml:C1RadialMenuItem Header="左へ挿入" IsCheckable="True" GroupName="MutuallyExclusiveGroup"/>
    <Xaml:C1RadialMenuItem Header="上へ挿入" DisplayIndex="2" IsCheckable="True"
GroupName="MutuallyExclusiveGroup"/>
    <Xaml:C1RadialMenuItem Header="右へ挿入" DisplayIndex="4" IsCheckable="True"
GroupName="MutuallyExclusiveGroup"/>
    <Xaml:C1RadialMenuItem Header="下へ挿入" DisplayIndex="6" IsCheckable="True"
GroupName="MutuallyExclusiveGroup" />
  </Xaml:C1RadialMenuItem>
</Xaml:C1RadialMenu>
```

相互に排他的なグループ内では、一度に1つの C1RadialMenuItem しかチェックできません。

ナビゲーションボタン

C1RadialMenu の**ナビゲーションボタン**は、ユーザーがアプリケーションを右タップしたときに最初に表示されるオブジェクトです。C1RadialMenu の **Icon** プロパティと **NavigationButtonRelativeSize** プロパティを使用して、**ナビゲーションボタン**をカスタマイズできます。また、デフォルトでは NavigationButtonRelativeSize プロパティを使用すると、C1RadialMenu のサイズに対して**ナビゲーションボタン**のサイズを設定することができます。デフォルトでは、このプロパティは 0.15 に設定されています。

たとえば、次は NavigationButtonRelativeSize を 0.15 に設定したデフォルトのナビゲーションボタンです。

NavigationButtonRelativeSize プロパティを 0.45 に設定すると、ナビゲーションボタンは次のようになります。

Icon プロパティを変更することもできます。デフォルトでは、Icon プロパティは「A」に設定されています。

デフォルトナビゲーションボタンの Icon プロパティを変更し、NavigationButtonRelativeSize プロパティを 0.25 に設定すると、次のようになります。

サブメニューのネスト

C1RadialMenu コントロールは、サブメニューを保持することができます。これらのサブメニューは、C1RadialMenuItem が他の C1RadialMenuItem タグ内にネストしていると作成されます。次に例を示します。

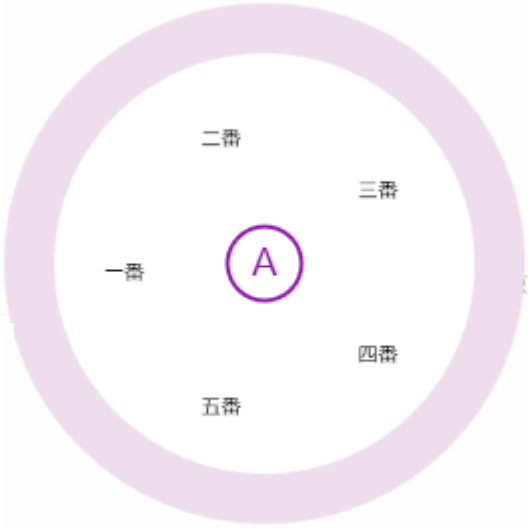
マークアップ

```
<c1:C1RadialMenuItem Header="一番">
    <c1:C1RadialMenuItem Header="二番">
        <c1:C1RadialMenuItem Header="三番">
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            <c1:C1RadialMenuItem Header="四番">
                <c1:C1RadialMenuItem Header="五番"/>
</c1:C1RadialMenuItem>
```

この XAML マークアップを別の C1RadialMenu の開始タグと終了タグの間に配置すると、次のようなサブメニューが作成されます。

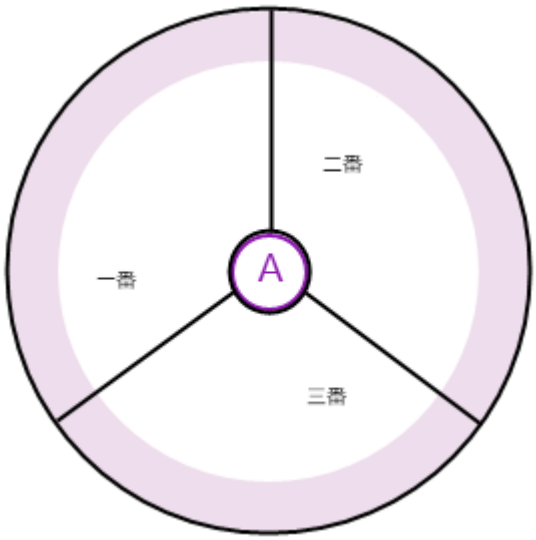


必要な数だけラジアルメニューをネストできますが、使いやすさの観点から、1つの階層にサブメニューを2、3個以上は置かない方がよいでしょう。

C1RadialMenu コントロールでネストしたサブメニューを作成するタスク別ヘルプについては、「サブメニューの作成」を参照してください。

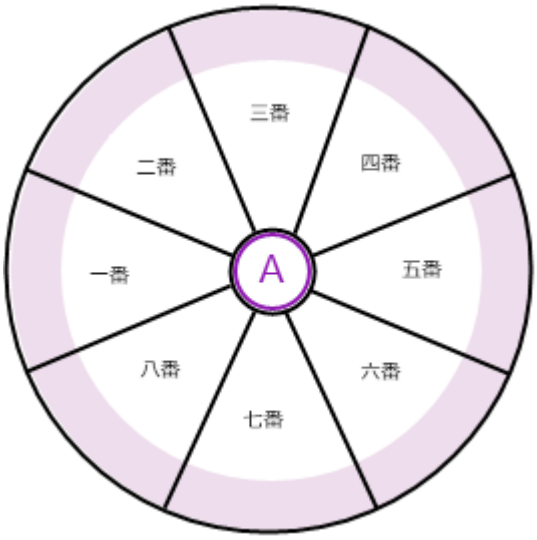
項目の配置

デフォルトでは、C1RadialMenuItem は 360 度のメニュー内に均等に配置されます。たとえば、**C1RadialMenu** に3つの項目がある場合は、各項目がそれぞれ C1RadialMenu の3分の1を占めるように配置されます。



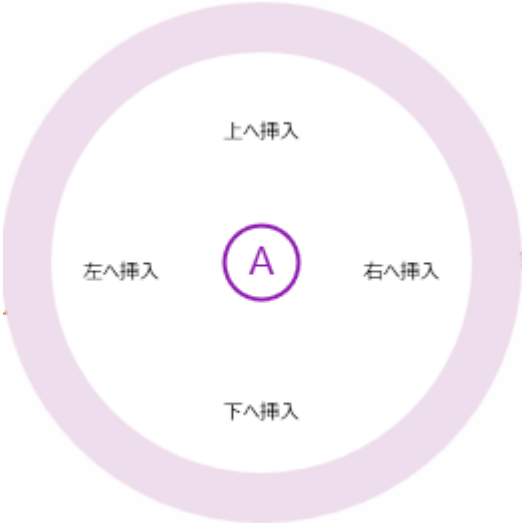
また、C1RadialMenu に8つの項目がある場合は、次の図のようになります。

Copyright © GrapeCity, Inc. All rights reserved.



各項目は、それぞれラジアルメニューの8分の1を占めています。各 **C1RadialMenuItem** は、1つのセクター内に配置されます。セクターの数は、**SectorCount** を使用してカスタマイズできます。このプロパティは、子項目を含むすべての C1RadialMenu および C1RadialMenuItem に設定できます。

SectorCount プロパティは、**DisplayIndex** プロパティと一緒に使用する場合、**C1RadialMenuItem** の配置は完全的にカスタマイズすることができます。DisplayIndex プロパティは C1RadialMenuItem の表示方法を定義するためにゼロベースのインデックスを使用しています。たとえば、SectorCount を「8」に設定した **C1RadialMenu** の表示インデックスは、次のようになります。



インデックスは C1RadialMenu の中心の左側からメニューに沿って時計回りに大きくなります。

たとえば、次の C1RadialMenu は、インデックス3とインデックス4の位置に2つの C1RadialMenuItem しか表示しません。

マークアップ

```
<Xaml:C1RadialMenu SectorCount=8>
   <Xaml:C1RadialMenuItem DisplayIndex="3" />
   <Xaml:C1RadialMenuItem DisplayIndex="4" />
</Xaml:C1RadialMenu>
```

SectorCount プロパティと DisplayIndex プロパティを使用して、4つの異なる位置に項目を表示することもできます。

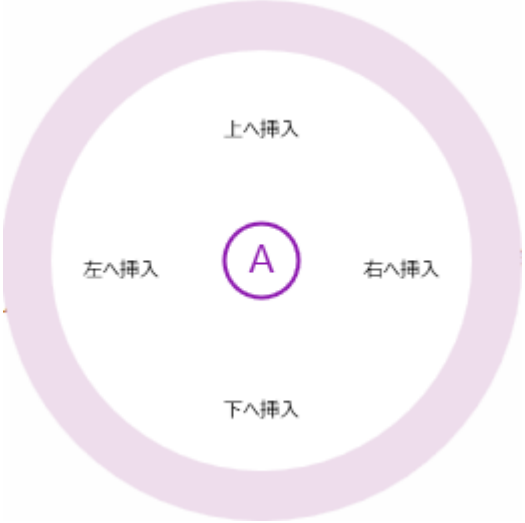
マークアップ

```
<Xaml:C1RadialMenu SectorCount="8">
  <Xaml:C1RadialMenuItem Header="左へ挿入"/>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
  <Xaml:C1RadialMenuItem Header="上へ挿入" DisplayIndex="2" />
  <Xaml:C1RadialMenuItem Header="右へ挿入" DisplayIndex="4" />
  <Xaml:C1RadialMenuItem Header="下へ挿入" DisplayIndex="6" />
</Xaml:C1RadialMenu>
```

上のマークアップは、次の図のような **C1RadialMenu** コントロールを作成します。



項目の選択

C1RadialMenuItem は、任意の数の子項目を持つことができます。**SelectedIndex** プロパティを設定して、選択された項目として表示する項目を制御できますが、デフォルトでは、**C1RadialMenuItem** はコレクションの最初の子項目を選択された項目として表示します。**AutoSelect** プロパティを設定しても、同じ設定になります。

C1RadialMenu コントロールは、**ShowSelectedItem** プロパティによって選択項目の記憶もサポートします。このプロパティを使用すると、事前に設定した項目ではなく、最後に選択された項目を表示することができます。ShowSelectedItem を使用する場合は、AutoSelect プロパティを True に設定する必要があります。

したがって、右揃えテキストをよく使用するユーザーがいると思われる場合は、左揃えオプションや中央揃えオプションをデフォルトとして使用するのではなく、最後に選択された項目を表示することができます。ShowSelectedItem プロパティを True に設定すると、事前に設定した項目や最初に選択された項目ではなく、ユーザーが最後に選択した項目が常に表示されます。

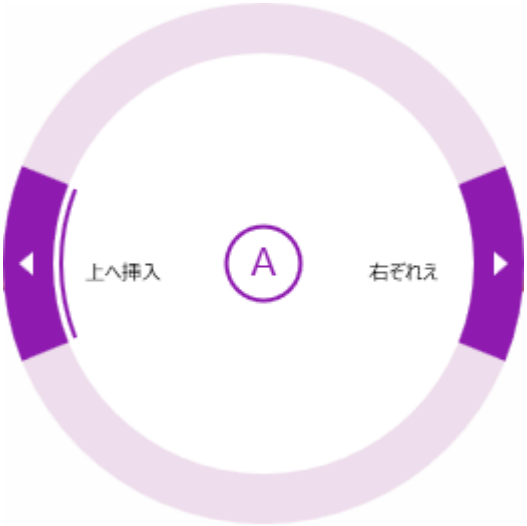
次のマークアップは、2つのサブメニューを含む簡単な C1RadialMenu を作成します。

マークアップ

```
<Xaml:C1RadialMenu SectorCount="8" >
  <Xaml:C1RadialMenuItem Header="挿入" SectorCount="8" AutoSelect="True" ShowSelectedItem="True" >
    <Xaml:C1RadialMenuItem Header="左へ挿入" />
    <Xaml:C1RadialMenuItem Header="上へ挿入" DisplayIndex="2" />
    <Xaml:C1RadialMenuItem Header="右へ挿入Insert Right" DisplayIndex="4" />
    <Xaml:C1RadialMenuItem Header="下へ挿入" DisplayIndex="6" />
  </Xaml:C1RadialMenuItem>
  <Xaml:C1RadialMenuItem Header="揃える" AutoSelect="True" SectorCount="8" DisplayIndex="4"
ShowSelectedItem="True">
    <Xaml:C1RadialMenuItem Header="右ぞろえ" />
    <Xaml:C1RadialMenuItem Header="左ぞろえ" />
    <Xaml:C1RadialMenuItem Header="中央ぞろえ" />
  </Xaml:C1RadialMenuItem>
</Xaml:C1RadialMenu>
```

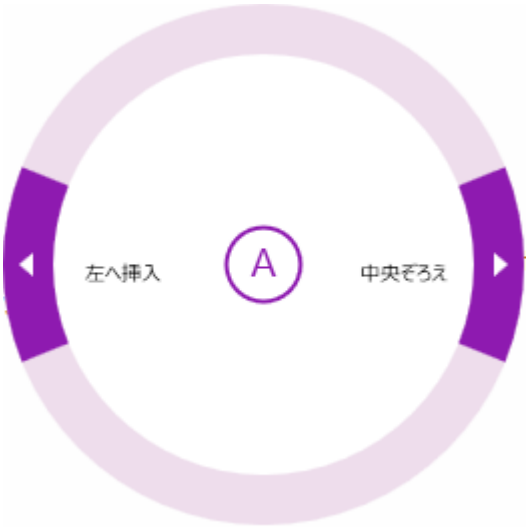

Copyright © GrapeCity, Inc. All rights reserved.

アプリケーションを実行すると、次の図のようになります。



[**Align Right**]オプションの上の**展開領域**をタップすると、次のメニューが表示されます。

[**Align Center**]をタップしてから、**戻る**矢印をタップします。メインメニューは、次の図のようになります。



[**Align Right**]が最後に選択された[**Align Center**]に置き換えられたことに注目してください。

レイアウトおよび外観

Copyright © GrapeCity, Inc. All rights reserved.

ClearStyle

Radial Menu for UWP は、コントロールのテンプレートを変更することなくコントロールの色を簡単に変更できる ComponentOne の ClearStyle 技術をサポートします。色のプロパティをいくつか設定するだけで、コントロール全体のスタイルを簡単に設定できます。

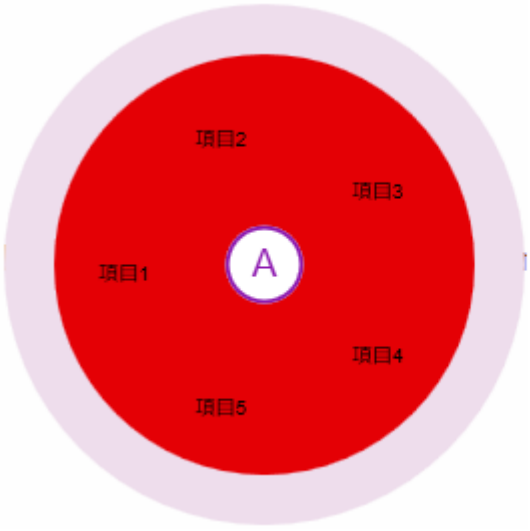
次の表に、**C1RadialMenu** コントロールのブラシのプロパティの概要を示します。

ブラシ	説明
AccentBrush	小さなメニューパーツに色を付けるブラシを取得または設定します。
Background	コントロールの背景のブラシを取得または設定します。
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。
Foreground	コントロールの前景を描画するブラシを取得または設定します。

次の表に、C1RadialMenuItem コントロールのブラシの概要を示します。

ClearStyle プロパティ	説明
AccentBrush	小さなメニューパーツに色を付けるブラシを取得または設定します。
Background	コントロールの背景のブラシを取得または設定します。
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。
Foreground	コントロールの前景を描画するブラシを取得または設定します。
HeaderForeground	ヘッダーの前景ブラシを取得または設定します。
HeaderBackground	ヘッダーの背景ブラシを取得または設定します。

いくつかのプロパティを設定することで、C1RadialMenu コントロールの外観を完全に変更できます。たとえば、**C1RadialMenu** の **Background** プロパティは、RadialMenu の背景色を設定します。**Background** プロパティを "#FFE40005" に設定すると、**C1RadialMenu** コントロールは次のようになります。



ComponentOne の ClearStyle 技術は、このように簡単に使用できます。

Copyright © GrapeCity, Inc. All rights reserved.

外観のプロパティ

Radial Menu for UWP には、コントロールの外観をカスタマイズするためのいくつかのプロパティがあります。コントロールに表示されるテキストの外観を変更したり、コントロールのグラフィック要素をカスタマイズすることができます。以下のトピックでは、これらの外観プロパティの一部について説明します。

テキストのプロパティ

次のプロパティを使用して、C1RadialMenu コントロールと C1RadialMenuItem 項目のテキストの外観をカスタマイズできます。

プロパティ	説明
FontFamily	コントロールのフォントファミリを取得または設定します。これは依存プロパティです。
FontSize	フォントサイズを取得または設定します。これは依存プロパティです。
FontStretch	フォントを画面上で伸縮する比率を取得または設定します。これは依存プロパティです。
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。
FontWeight	指定されたフォント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。

色のプロパティ

次のプロパティを使用して、C1RadialMenu コントロールと C1RadialMenuItem 項目で使用す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、C1RadialMenu コントロールと C1RadialMenuItem 項目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して、C1RadialMenu コントロールと C1RadialMenuItem 項目のサイズをカスタマイズできます。

プロパティ	説明
ExpandAreaThickness	展開領域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。
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。

Copyright © GrapeCity, Inc. All rights reserved.

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。

タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio .NET でのプログラミングに精通しており、**C1RadialMenu** 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。**Radial Menu for UWP** 製品を初めて使用される場合は、まず「**Radial Menu for UWP** 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、**C1RadialMenu** コントロールを使用して特定のタスクを実行するためのソリューションを提供します。

また、タスク別ヘルプトピックは、新しい Windows ストアプロジェクトが既に作成されていることを前提としています。

ラジアルメニューの作成

最上位のメニューの作成

このトピックでは、**C1RadialMenu** コントロールの最上位のラジアルメニューを作成する方法について学習します。

XAML の場合

次の手順に従います。

1. `<Grid>` タグと `</Grid>` タグの間に次の XAML を配置します。

マークアップ

```
<Xaml:C1ContextMenuService.ContextMenu>
    <Xaml:C1RadialMenu >
   </Xaml:C1RadialMenu>
 </Xaml:C1ContextMenuService.ContextMenu>
```

2. 次の XAML を `<Xaml:C1RadialMenu>` タグと `</Xaml:C1RadialMenu>` タグの間に配置します。

マークアップ

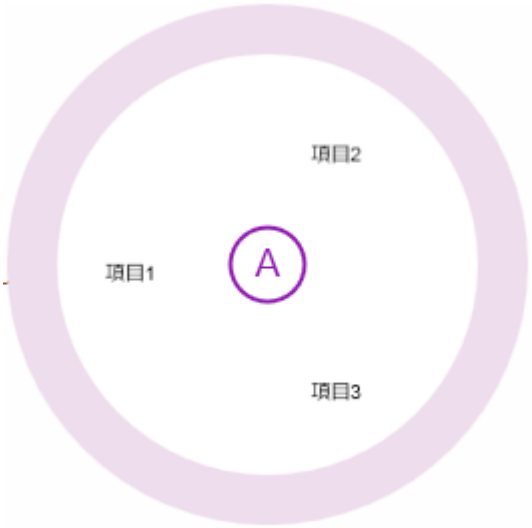
```
<Xaml:C1RadialMenuItem Header="項目1" />
<Xaml:C1RadialMenuItem Header="項目2" />
<Xaml:C1RadialMenuItem Header="項目3" />
```

3. プログラムを実行し、次の点を確認します。
   - ページを右タップまたは右クリックします。この場合は、これが **C1RadialMenu** がアタッチされている要素です。ナビゲーションボタンが表示されることを確認します。
   - ラジアルメニューを表示するには、ナビゲーションボタンをタップまたはクリックします。C1RadialMenu に3つの C1RadialMenuItem が含まれていることを確認します。

このトピックの作業結果

このトピックの作業結果は次のようになります。

Copyright © GrapeCity, Inc. All rights reserved.



サブメニューの作成

このトピックでは、C1RadialMenu の項目の1つにアタッチされるサブメニューを作成します。このトピックでは、少なくとも1つの **C1RadialMenuItem** を持つ最上位のラジアルメニューが既に作成されていることを前提としています(「最上位のメニューの作成」を参照)。

次の手順に従います。

1. `<Xaml:C1RadialMenu>` タグと `</Xaml:C1RadialMenu>` タグの間に次の XAML を配置します。

マークアップ

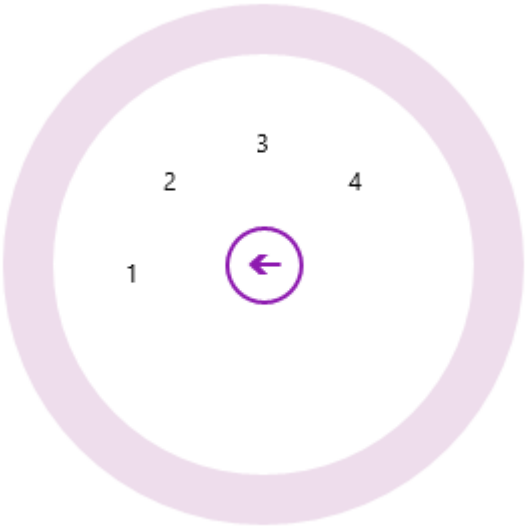
```
<Xaml:C1RadialMenuItem Header="タップ" >
  <Xaml:C1RadialMenuItem Header="項目１" />
  <Xaml:C1RadialMenuItem Header="項目２" />
  <Xaml:C1RadialMenuItem Header="項目３" />
</Xaml:C1RadialMenuItem>
```

2. プログラムを実行します。
   - ページを右タップまたは右クリックして、ナビゲーションボタンを表示します。ナビゲーションボタンをタップまたはクリックして、メイン **C1RadialMenu** を表示します。
   - [**Tap Here**]項目をタップし、作成したサブメニューが表示されることを確認します。

このトピックの作業結果

次は、[**Tap Here**]項目をタップした後の **C1RadialMenu** です。

Copyright © GrapeCity, Inc. All rights reserved.



ColorPicker メニューの作成

C1RadialColorItem を使用して、C1RadialMenu コントロールを使用するカラーピッカーを作成することができます。このトピックでは、C1RadialMenu アプリケーションを作成し、いくつかの C1RadialColorItem を C1RadialMenu コントロールに追加します。XAML マークアップとコードの両方を使用してアプリケーションを作成します。

XAML ビュー

1. ページ内の `<Grid> </Grid>` タグを見つけ、このタグを次のマークアップに置き換えて、アプリケーションのフレームワークを作成します。

マークアップ

```
<Grid Background="{ThemeResourceApplicationPageBackgroundThemeBrush}">
      <Border Background="LemonChiffon" MinHeight="40"BorderBrush="#969696"
            BorderThickness="1"Padding="5" HorizontalAlignment="Stretch"VerticalAlignment="Stretch">
        <Xaml:C1ContextMenuService.ContextMenu>

        </Xaml:C1ContextMenuService.ContextMenu>

       <Xaml:C1ListViewer x:Name="text"Foreground="Sienna"
HorizontalAlignment="Stretch"VerticalAlignment="Center" Height="75"ZoomMode="Disabled"
Xaml:C1NagScreen.Nag="True">
            <TextBlock Text="Touch:hold down for a few seconds until the indicator
displays.&#10;Keyboard:press the context-menu button over this text.&#10;Mouse: right-click overthis text."
                  FontSize="16" TextWrapping="Wrap" />
        </Xaml:C1ListViewer>
      </Border>
      <TextBlock x:Name="txt" Foreground="Red"Text="" FontSize="16"
VerticalAlignment="Bottom"HorizontalAlignment="Center" Margin="10" />
  </Grid>
```

2. 追加したマークアップ内にある `<Xaml:C1ContextMenuService.ContextMenu> </Xaml:C1ContextMenuService.ContextMenu>` タグを探します。このタグの間にカーソルを置きます。
3. Visual Studio ツールボックスの **C1RadialMenu** コントロールを見つけてダブルクリックし、このコントロールをアプリケーションに追加します。
4. 次のように、開始タグ `<C1RadialMenu>` を編集します。

Copyright © GrapeCity, Inc. All rights reserved.

マークアップ

```
<Xaml:C1RadialMenux:Name="contextMenu" Offset="-130,0"ItemClick="contextMenu_ItemClick" >
```

5. 次のマークアップを <C1RadialMenu></C1RadialMenu> タグの間に追加して、3つの C1RadialColorItem をアプリケーションに追加します。

マークアップ

```
<Xaml:C1RadialColorItemx:Name="rainbowItem" ShowSelectedItem="True"
AutoSelect="True"IsSelectable="False" >

</Xaml:C1RadialColorItem>

<Xaml:C1RadialColorItemx:Name="greenItem" SelectedIndex="5"ShowSelectedItem="True"
AutoSelect="True"IsSelectable="False" >

</Xaml:C1RadialColorItem>

<Xaml:C1RadialColorItemx:Name="blueItem" SelectedIndex="0"ShowSelectedItem="True"
AutoSelect="True"IsSelectable="False" >

</Xaml:C1RadialColorItem>
```

6. 「rainbowItem」という名前の C1RadialColorItem を選択し、次のマークアップを開始タグと終了タグの間に挿入します。これで、SolidColorBrush のサブメニュー項目が追加されます。

マークアップ

```
<SolidColorBrushColor="Red"/>
<SolidColorBrushColor="Orange"/>
<SolidColorBrushColor="Yellow"/>
<SolidColorBrushColor="Green"/>
<SolidColorBrushColor="Blue"/>
<SolidColorBrushColor="Indigo"/>
<SolidColorBrushColor="Violet"/>
```

7. 「greenItem」という名前の C1RadialColorItem を選択し、次のマークアップを開始タグと終了タグの間に挿入します。これで、いくつかの階調の緑色に対応する C1RadialColorItem サブメニュー項目が追加されます。

マークアップ

```
<Xaml:C1RadialColorItemx:Name="greenItem" SelectedIndex="5"
                        ShowSelectedItem="True" AutoSelect="True"IsSelectable="False"
Xaml:C1NagScreen.Nag="True">
                <Xaml:C1RadialColorItem ToolTip="Lime"Brush="#FF92D050" GroupName="Green"/>
                <Xaml:C1RadialColorItem ToolTip="Light Green"Brush="#FFC6EFCE" GroupName="Green"/>
                <Xaml:C1RadialColorItem ToolTip="Green"Brush="#FF00FF00" GroupName="Green"/>
                 <Xaml:C1RadialColorItemToolTip="Dark Green" Brush="#FF1D421E"GroupName="Green"/>
                <Xaml:C1RadialColorItem ToolTip="Dark Green"Brush="#FF1D5A2D" GroupName="Green"/>
                <Xaml:C1RadialColorItem ToolTip="Dark Green"Brush="Green" GroupName="Green"/>
                <Xaml:C1RadialColorItem ToolTip="Dark Green"Brush="#FF008000" GroupName="Green"/>
                <Xaml:C1RadialColorItem ToolTip="Dark Green"Brush="#FF00B050" GroupName="Green"/>
              </Xaml:C1RadialColorItem>
```

8. 「blueItem」という名前の C1RadialColorItem を選択し、次のマークアップを開始タグと終了タグの間に挿入します。これで、いくつかの階調の青色に対応する C1RadialColorItem サブメニュー項目が追加されます。

マークアップ

```
<Xaml:C1RadialColorItemx:Name="blueItem" SelectedIndex="0"
```


Copyright © GrapeCity, Inc. All rights reserved.

```
                        ShowSelectedItem="True" AutoSelect="True"IsSelectable="False"
Xaml:C1NagScreen.Nag="True">
                <Xaml:C1RadialColorItem ToolTip="Blue"Brush="Blue" GroupName="Blue"/>
                <Xaml:C1RadialColorItem ToolTip="Slate Blue"Brush="MediumSlateBlue"
GroupName="Blue"/>
                <Xaml:C1RadialColorItem ToolTip="Turquoise"Brush="Turquoise" GroupName="Blue"/>
                 <Xaml:C1RadialColorItemToolTip="Aqua" Brush="Aqua" GroupName="Blue"/>
                <Xaml:C1RadialColorItem ToolTip="Sky Blue"Brush="SkyBlue" GroupName="Blue"/>
                <Xaml:C1RadialColorItem ToolTip="Purple"Brush="#FFAC38AC"
AccentBrush="#FF801C80"GroupName="Blue"/>
                <Xaml:C1RadialColorItem ToolTip="Dark Purple"Brush="Purple" GroupName="Blue"/>
                <Xaml:C1RadialColorItem ToolTip="Dark Blue"Brush="DarkBlue" GroupName="Blue"/>
              </Xaml:C1RadialColorItem>
```

9. ページを右クリックし、リストから[**コードの表示**]を選択します。
10. 次の using 文をページの先頭に追加します。

C# コードの書き方

C#

```
usingC1.Xaml;
```

11. 次の **ItemClick** イベントをページに追加します。これで、C1RadialMenu から色を選択した際に、メインページに追加したテキストの色を変化させることができます。

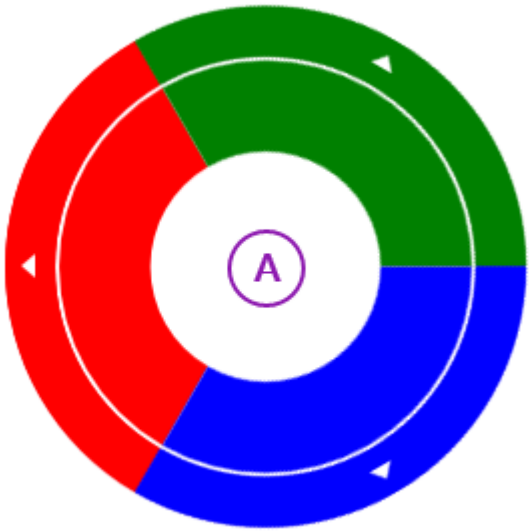
C# コードの書き方

C#

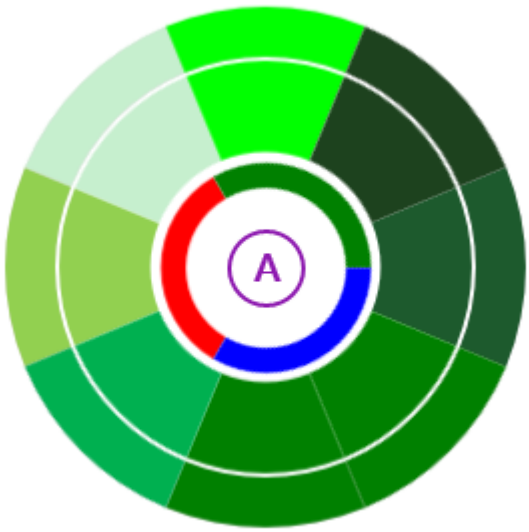
```
private voidcontextMenu_ItemClick(object sender, SourcedEventArgs e)
    {
        C1RadialMenuItem item = e.Source asC1RadialMenuItem;
        if (item is C1RadialColorItem)
        {
            this.text.Foreground =((C1RadialColorItem)item).Brush;
            txt.Text = "選択されました:" + ((C1RadialColorItem)item).Color.ToString();
        }
        else
        {
            txt.Text = "選択されました:" +(item.Header ?? item.Name).ToString();
        }
    }
```

12. [F5]キーを押すか、デバッグを開始して、アプリケーションを実行します。アプリケーションを右クリックまたは右タップして C1RadialMenu を開くと、次の図のように表示されます。

Copyright © GrapeCity, Inc. All rights reserved.



13. 緑色のサブメニューを選択すると、C1RadialMenu は次の図のようになります。



内側の色を選択することで、メインメニューに戻ることができます。

14. 色を選択します。アプリケーションページ内のテキストの色が変わることがわかります。

数値ラジアルメニューの作成

C1RadialMenu コントロールを使用して、円形の数値スライダメニューを作成できます。このメニューは、特に、ユーザーにフォントサイズを選択してもらうアプリケーションで役立ちます。アプリケーションの作成には、XAML マークアップとコードの両方を使用します。

1. ページ内の `<Grid> </Grid>` タグを見つけ、このタグを次のマークアップに置き換えて、アプリケーションのフレームワークを作成します。

XAML でマークアップの書き方

マークアップ

Copyright © GrapeCity, Inc. All rights reserved.

```
<Page.Resources>
      <Style TargetType="TextBlock" x:Key="TextIconStyle">
         <Setter Property="Margin" Value="-10" />
         <Setter Property="FontSize" Value="20" />
         <Setter Property="FontFamily" Value="Segoe UI Symbol" />
         <Setter Property="FontWeight" Value="Normal" />
         <Setter Property="Foreground" Value="#333333" />
         <Setter Property="HorizontalAlignment" Value="Center" />
         <Setter Property="VerticalAlignment" Value="Center" />
      </Style>
      <Style TargetType="Image" x:Key="MenuIcon">
         <Setter Property="Width" Value="16"/>
         <Setter Property="Height" Value="16"/>
         <Setter Property="Margin" Value="0"/>
      </Style>
   </Page.Resources>

   <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
      <Xaml:C1ContextMenuService.ContextMenu>

      </Xaml:C1ContextMenuService.ContextMenu>

      <Xaml:C1ListViewer x:Name="text" Foreground="Sienna" HorizontalAlignment="Stretch"
VerticalAlignment="Center" Height="75" ZoomMode="Disabled" FontSize="16"
Xaml:C1NagScreen.Nag="True">
         <TextBlock Text="タッチ: インジケータが表示されるまでに数秒間押したままにします。&#10;キーボード: このテキスト上にあるコンテキストメニューボタンを押してください。&#10;マウス: このテキスト上で右クリックします。"
                    TextWrapping="Wrap" />
      </Xaml:C1ListViewer>

      <TextBlock x:Name="txt" Foreground="Red" Text="" FontSize="16" VerticalAlignment="Bottom"
HorizontalAlignment="Center" Margin="10" />

   </Grid>
```

2. 追加したマークアップにある `<Xaml:C1ContextMenuService.ContextMenu> </Xaml:C1ContextMenuService.ContextMenu>` タグを探します。このタグの間にカーソルを置きます。
3. Visual Studio ツールボックスで **C1RadialMenu** コントロールを見つけてダブルクリックし、このコントロールをアプリケーションに追加します。
4. 次のように、開始タグ `<C1RadialMenu>` を編集します。

マークアップ

```
<Xaml:C1RadialMenu Xaml:C1NagScreen.Nag="True" x:Name="contextMenu" Offset="-130,0"
Opened="contextMenu_Opened" AccentBrush="ForestGreen"
                   ItemClick="contextMenu_ItemClick" ItemOpened="contextMenu_ItemOpened"
BorderBrush="#FFC6DEC4">
```

5. 次のマークアップを `<C1RadialMenu> </C1RadialMenu>` タグの間に追加して、1つの **C1RadialNumericItem** をアプリケーションに追加します。同時に、いくつかのサブ項目と1つのカスタム項目アイコンも追加します。

XAML でマークアップの書き方

マークアップ

```
<Xaml:C1RadialNumericItem Header="Font Size" Minimum="9" Maximum="72" MarkStartAngle="-128"
```


Copyright © GrapeCity, Inc. All rights reserved.

```
MarkEndAngle="231" x:Name="fontSize" Value="11">
            <Xaml:C1RadialNumericItem.Icon>
              <TextBlock Style="{StaticResource TextIconStyle}" FontFamily="Segoe UI" FontSize="18">
                    <Run Text="A"/>
                    <Run Text="{Binding Value, ElementName=fontSize}"
Typography.Variants="Superscript"/>
              </TextBlock>
            </Xaml:C1RadialNumericItem.Icon>
            <x:Double>9</x:Double>
            <x:Double>11</x:Double>
            <x:Double>13</x:Double>
            <x:Double>16</x:Double>
            <x:Double>20</x:Double>
            <x:Double>36</x:Double>
            <x:Double>72</x:Double>
          </Xaml:C1RadialNumericItem>
```

6. ページを右クリックし、リストから**[コードの表示]**を選択します。
7. 次の using 文をページの先頭に追加します。

C#

```
using C1.Xaml;
```

8. 次の **ItemClick** イベントをページに追加します。これで、選択した項目に合わせて、TextBox コントロール内のテキストのサイズを変更できるように なります。

C# コードの書き方

C#

```
private void contextMenu_ItemClick(object sender, SourcedEventArgs e)
        {
            C1RadialMenuItem item = e.Source as C1RadialMenuItem;

if (item is C1RadialNumericItem)
            {
                txt.FontSize = ((C1RadialNumericItem)item).Value;
                txt.Text = "クリックされた項目： " +
((C1RadialNumericItem)item).Value.ToString();
            }
            else
            {
                txt.Text = "クリックされた項目： " + (item.Header ??
item.Name).ToString();
            }
        }
```

9. さらに、2つのイベントを追加します。このコードは、ItemOpened イベントと Opened イベントを制御します。

C# コードの書き方

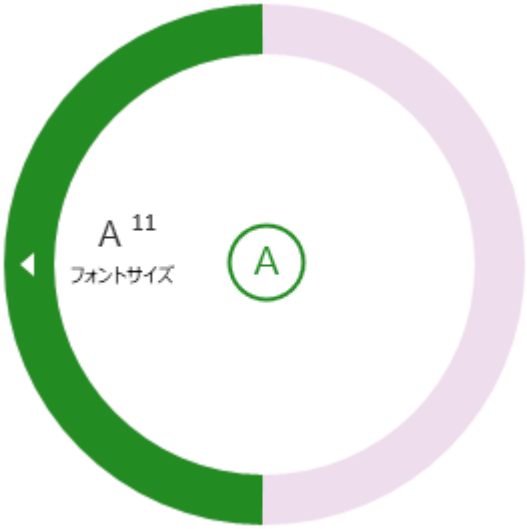
C#

```
private void contextMenu_ItemOpened(object sender, SourcedEventArgs e)
        {
            C1RadialMenuItem item = e.Source as C1RadialMenuItem;
            txt.Text = "開かれた項目： " + (item.Header ?? item.Name).ToString();
```

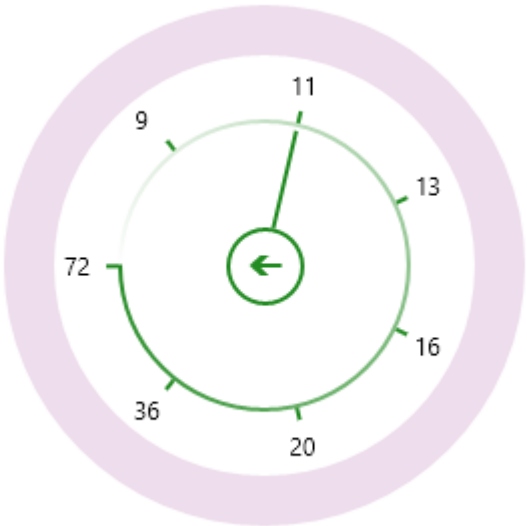

Copyright © GrapeCity, Inc. All rights reserved.

```
        }
        private void contextMenu_Opened(object sender, EventArgs e)
        {
            // このサンプルでは、編集可能な基底のコントロールがないため、 即座にメニューを展開します
            contextMenu.Expand();
        }
```

10. [F5]キーを押すか、デバッグを開始して、アプリケーションを実行します。**C1RadialMenu** コントロールは次の図のようになります。



11. [Font Size]オプションをクリックすると、**C1NumericRadialItem** が表示されます。



チェック可能なラジアルメニュー項目の使い方

次のトピックでは、スタンドアロンのメニュー項目、および相互に排他的なチェック可能なラジアルメニュー項目を作成する方法について学習します。

チェック可能な C1RadialMenuItem の作成

このトピックでは、ユーザーがオンまたはオフにすることができるチェック可能な C1RadialMenuItem を作成します。このトピックを完了するには、1つ以上の項目を持つ **C1RadialMenu** コントロールか、1つ以上のサブメニューを持つ C1RadialMenu コントロールが必要です。

Copyright © GrapeCity, Inc. All rights reserved.

XAML の場合

次の手順に従います。

1. チェック可能にする **C1RadialMenuItem** の `<Xaml:C1RadialMenuItem>` タグを探し、タグに `IsCheckable="True"` を追加します。XAML は次のようになります。

マークアップ
<Xaml:C1RadialMenuItem Header="C1RadialMenuItem" IsCheckable="True"/>

2. プロジェクトを実行します。

コードの場合

次の手順に従います。

1. ソースビューで、チェック可能にする項目の `<Xaml:C1RadialMenuItem>` タグを探し、それに `Name="CheckableRadialMenuItem"` を追加します。これにより、コードから使用できる一意の識別子を項目に渡すことができます。
2. コードビューに切り替えて、**InitializeComponent()** メソッドの下に次のコードを追加します。

Visual Basic コードの書き方

Visual Basic
CheckableRadialMenuItem.IsCheckable = True

C# コードの書き方

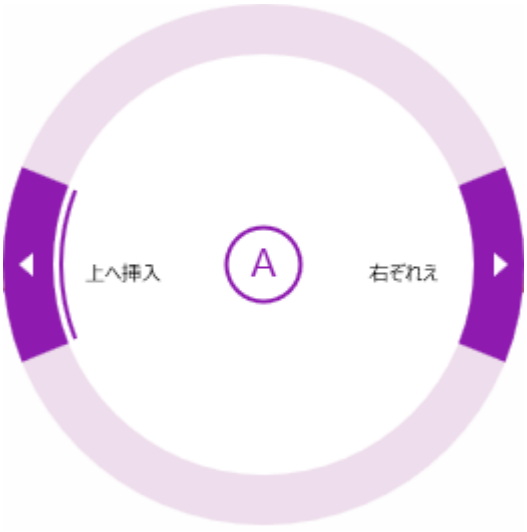
C#
CheckableRadialMenuItem.IsCheckable = true;

3. プログラムを実行します。

このトピックの作業結果

プログラムを実行したら、**C1RadialMenu** を開きます。チェックマークを付けるには、**C1RadialMenuItem** をクリックします。

下の画像はチェック可能な **C1RadialMenuItem** を示しています。



Copyright © GrapeCity, Inc. All rights reserved.

相互に排他的なチェック可能なラジアルニュー項目の作成

このトピックでは、一度に1つの項目のみをオンにできるようにグループ化されたチェック可能な C1RadialMenuItem のリストを作成する方法を学習します。

XAML の場合

次の手順に従います。

1. 相互に排他的なチェック可能な項目のグループに追加する各 **C1RadialMenuItem** の `<c1:C1RadialMenuItem>` タグに、`IsCheckable="True"` と `GroupName="CheckableGroup"` を追加します。
2. プログラムを実行してグループの1番目の項目をクリックします。C1RadialMenuItem が強調表示されていることを確認します。ここで、グループの2番目の項目をクリックします。1番目の項目から強調表示が削除され、2番目の項目に追加されることを確認します。

コードの場合

次の手順に従います。

1. 相互に排他的なチェック可能な項目のグループに追加する各 C1RadialMenuItem 項目の **Name** プロパティを設定します。
2. **MainPage.xaml.cs** ページを開きます。
3. 各 **C1RadialMenuItem** の IsCheckable と GroupName プロパティを設定し、"ItemName" を **C1RadialMenuItem** の **Name** プロパティの値に置き換えます。

   Visual Basic コードの書き方

   ```
   Visual Basic
   ItemName.IsCheckable = True
   ```

   C# コードの書き方

   ```
   C#
   ItemName.IsCheckable = true;
   ```

4. プログラムを実行してグループの1番目の項目をタップします。
   - 強調表示の線を細くしたようなチェックが C1RadialMenuItem に追加されることを確認します。
   - ここで、グループの2番目の項目をタップします。1番目の項目からチェックが削除され、2番目の項目に追加されることを確認します。

C1RadialMenu の外観をカスタマイズする

いくつかのプロパティを使用するだけで、C1RadialMenu の外観をすばやく簡単にカスタマイズできます。

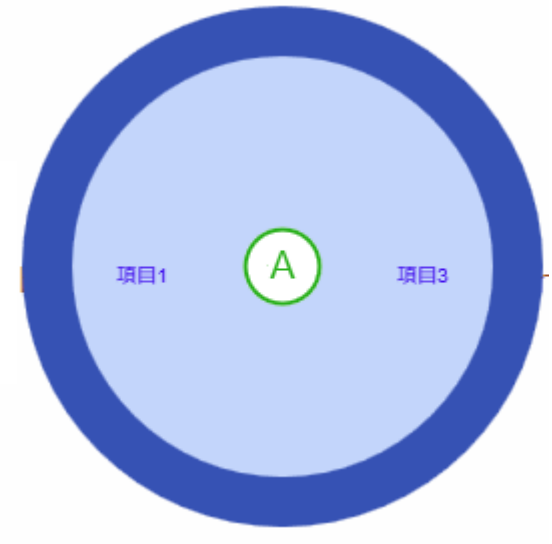
次の手順に従います。

1. 次のプロパティを `<Xaml: C1RadialMenu>` タグに追加します。
   - AccentBrush="#FF28B01A"
   - Background="#FFC3D5FB"
   - BorderBrush="#FF3652B4"
   - Foreground="#FF4210EE"
2. アプリケーションを実行します。

このトピックの作業結果

Copyright © GrapeCity, Inc. All rights reserved.

アプリケーションを実行すると、次の図のようになります。



自動メニュー折りたたみを有効にする

自動メニュー折りたたみ機能により、ユーザーはメニュー領域以外の場所をクリックして **C1RadialMenu** を折りたたむことができます。自動メニュー折りたたみを有効にするには、**AutoCollapse** プロパティを **True** に設定します。

XAML の場合

次の手順に従います。

1. `AutoCollapse="True"` を `<Xaml:C1RadialMenu>` タグに追加します。
2. プログラムを実行します。

コードの場合

次の手順に従います。

1. ソースビューで、チェック可能にする項目の `<Xaml:C1RadialMenuItem>` タグを探し、それに `Name="C1RadialMenu1"` を追加します。これにより、コードから使用できる一意の識別子を項目に渡すことができます。
2. コードビューに切り替えて、**InitializeComponent()** メソッドの下に次のコードを追加します。

   Visual Basic コードの書き方

   ```
   C1RadialMenu1.AutoCollapse = True
   ```

   C# コードの書き方

   ```
   C1RadialMenu1.AutoCollapse = true;
   ```

3. プログラムを実行します。

このトピックの作業結果

プロジェクトを実行したら、ページを右タップしてナビゲーションボタンを表示します。ナビゲーションボタンをタップして、

Copyright © GrapeCity, Inc. All rights reserved.

C1RadialMenu を開きます。ラジアルメニューを開いた状態で、メニュー以外の場所をクリックして、**C1RadialMenu** が閉じることを確認します。

ラジアルメニュー項目間へのセパレータの加

このトピックでは、ラジアルメニュー項目間にセパレータを追加する方法について学習します。セパレータは、2つの C1RadialMenuItem の間に空のセクターとして表示されます。C1RadialMenuItem 間にセパレータを追加する方法は2通りあります。それには、既存のメニュー項目の間に空の C1RadialMenuItem を配置するか、各 C1RadialMenuItem の **C1RadialMenu SectorCount** および DisplayIndex を設定します。

XAML の場合

次のいずれかを実行します。

C1RadialMenuItem の挿入

他のメニュー項目の間に空の C1RadialMenu を配置してセパレータを追加するには、2つの `<Xaml:C1RadialMenuItem>` タグの間に `<Xaml:C1RadialMenuItem />` を配置します。

マークアップ

```
<Xaml:C1RadialMenu SectorCount="8" >
      <Xaml:C1RadialMenuItem Header="項目1" />
      <Xaml: C1RadialMenuItem />
      <Xaml:C1RadialMenuItem Header="項目2"  />
      <Xaml: C1RadialMenuItem />
      <Xaml:C1RadialMenuItem Header="項目3"  />
    </Xaml:C1RadialMenu>
```

SectorCount と DisplayIndex の設定

各 C1RadialMenuItem の **C1RadialMenu** SectorCount および DisplayIndex を設定して、アプリケーションにセパレータを追加する場合、XAML マークアップは次のようになります。

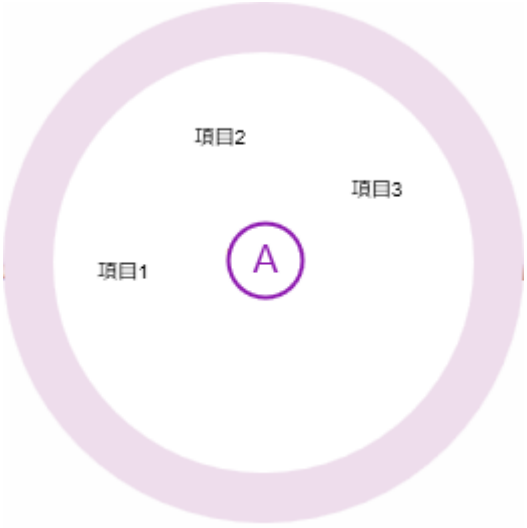
マークアップ

```
<Xaml:C1RadialMenu SectorCount="8" >
   <Xaml:C1RadialMenuItem Header="項目1" />
   <Xaml:C1RadialMenuItem Name="C1RadialMenuItem1" Header="項目2"
                  DisplayIndex="2" />
   <Xaml:C1RadialMenuItem Header="項目3" DisplayIndex="4" />
</Xaml:C1RadialMenu>
```

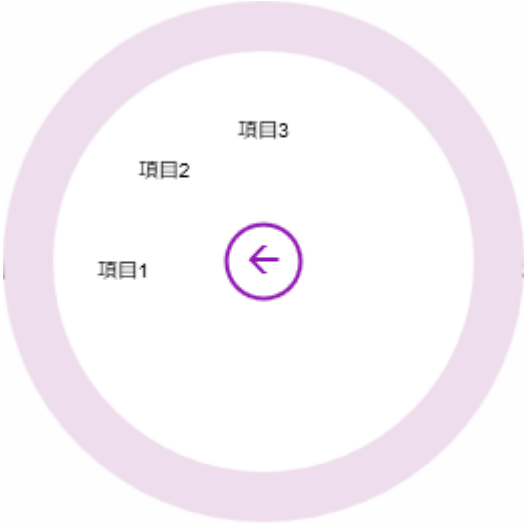
このトピックの作業結果

どちらのマークアップサンプルでも、結果は次の図のようになります。

Copyright © GrapeCity, Inc. All rights reserved.



比較のために、同じマークアップでセパレータを追加しない場合の結果は、次の図のような C1RadialMenu になります。



RadialMenu 項目へのアイコンの追加

この手順では、C1RadialMenuItem にアイコンを追加する方法について説明します。

XAML の場合

次の手順に従います。

1. Windows ストアプロジェクトにアイコン画像を追加します。12x12 ピクセルの画像が最適です。
2. `<Xaml:C1RadialMenuItem>` タグと `</Xaml:C1RadialMenuItem>` タグの間に、次の XAML マークアップを追加し、Source プロパティを画像の名前に置き換えます。

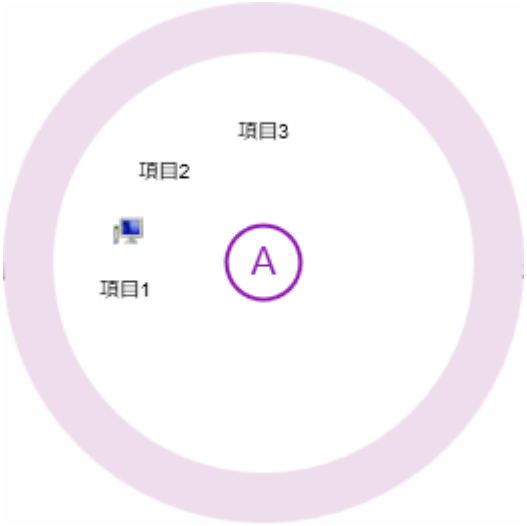
マークアップ
<Xaml:C1RadialMenuItem.Icon>   <Image Source="YourImage.png" Height="12" Width="12" Margin="5,0,0,0"/> </Xaml:C1RadialMenuItem.Icon>

3. プロジェクトを実行します。

このトピックの作業結果

次の図は、12x12 ピクセルのアイコン付きの C1RadialMenuItem です。

Copyright © GrapeCity, Inc. All rights reserved.



TabControl for UWP

TabControl for UWP を使用すると、コンテンツをタブとして整理し、それらのタブ間を移動することができます。タブは、空きスペースを有効利用して、選択可能な項目をすべてユーザーに表示するために役立ちます。タブはページの上下左右に配置でき、数種類の形状と組み込み機能がサポートされています。

主な特長

TabControl 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。**TabControl for UWP** は、次の主な特長を備えています。

- **モダンなタブスタイル**
  組み込まれている Rounded、Rectangle、Ribbon、Sloped の4種類の形状のいずれかを使用して、タブヘッダーの形状を変更できます。あるいは、輪郭の形状を使用せずに、すっきりしたモダンな外観にすることもできます。
- **任意の端へのタブの配置**
  タブは、上下左右のいずれにも配置できます。それには、**TabStripPlacement** プロパティを設定するだけです。
- **タブの重なり**
  タブ項目ヘッダー間の重なりをカスタマイズすることにより、Microsoft Visual Studio の[ドキュメント]タブのような階段状のタブを表示できます。それには、**TabStripOverlap** プロパティを設定するだけです。**TabStripOverlapDirection** プロパティを使用して、タブ項目を背面の最右端または背面の最左端で重ねるかどうかを定義します。選択された項目は常に最前面に置かれます。
- **タブを閉じる**
  ユーザーがタブを閉じることができるかどうか、および閉じるボタンを表示する場所を制御します。Visual Studio の[ドキュメント]タブと同様に、閉じるボタンを各タブ項目内またはタブストリップ外のグローバルな場所に表示できます。
- **メニュータブ**
  すべての項目を1つのメニューに表示することができます（Visual Studio の[ドキュメント]タブと同様）。これは、使用可能なスペースに項目が収まらない場合でも、エンドユーザーがすべての要素にすばやくアクセスできるため便利です。
- **要素のスクロール**
  C1TabControl は、現在使用可能なスペースにタブ項目が収まらない場合に、Microsoft Internet Explorer のような[前へ]/[後へ]ボタンを表示します。

クイックスタート

このクイックスタートガイドは、**TabControl 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、**C1TabControl** コントロールを含む新しいプロジェクトを作成します。また、C1TabControl コントロールをカスタマイズし、コンテンツを挿入したタブページを追加してから、コントロールの実行時機能をいくつか使用してみます。

Copyright © GrapeCity, Inc. All rights reserved.

手順1：C1TabControl アプリケーションの作成

この手順では、**TabControl for UWP** を使用して、Windows ストアアプリケーションを作成します。

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、言語の下で[**Windows ストア**]を選択し、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. ソリューションエクスプローラでプロジェクト名を右クリックし、[**参照の追加**]を選択します。
4. [**参照マネージャ**]ダイアログボックスで[**ComponentOne for UWP**]を選択します。[**OK**]をクリックしてダイアログボックスを閉じ、参照を追加します。
5. **MainPage.xaml** をまだ開いていない場合は開き、`<Page>` タグ内に次のマークアップを追加します。

   マークアップ

   ```
   xmlns:c1="using:C1.Xaml"
   ```

   これにより、C1.Xaml アセンブリへの参照がプロジェクトに追加されます。

これで、**TabControl for UWP** クイックスタートの最初の手順は終了です。この手順では、Windows ストアプロジェクトを作成しました。次の手順では、コントロールにタブとタブページを追加します。

手順2：アプリケーションへの TabControlの追加

前の手順では、Windows ストアプロジェクトを作成しました。この手順ではアプリケーションに **C1TabControl** を追加します。

次の手順に従います。

1. **MainPage.xaml** で、`<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。
2. `<Grid>` タグと `</Grid>` タグの間に次のマークアップを追加します。

   マークアップ

   ```
   <c1:C1TabControl x:Name="tabControl" TabStripOverlapDirection="Left"  TabStripOverlap="8"
   TabItemShape="Sloped" TabItemClose="InEachTab" TabStripPlacement="Bottom"
   TabStripMenuVisibility="Visible">

   </c1:C1TabControl>
   ```

   このマークアップは、プロジェクトに **C1TabControl** を追加します。また、プロパティは以下のように設定されています。
   - **TabStripOverlapDirection** プロパティは **Left** に、**TabStripOverlap** は **8** に設定されています。これは、タブ間で許可される重なりを示します。
   - **TabItemClose** プロパティは **InEachTab** に設定されています。これにより、各タブに閉じるボタンが追加されます。
   - **TabItemShape** プロパティは **Sloped** に設定されています。これにより、タブ項目の形状が Office フォルダのタブに似せて変更されます。
   - **TabStripMenuVisibility** プロパティは **Visible** に設定されています。
   - **TabStripPlacement** プロパティは **Bottom** に設定されています。これにより、タブストリップがコントロールの下端に配置されます。
     また、**C1TabControl** 内に **C1TabItem** を追加する必要があります。
3. `<c1:C1TabControl>` タグと `</c1:C1TabControl>` タグの間に次のマークアップを追加します。

   マークアップ

   ```
   <c1:C1TabItem Header="Notes" CanUserPin="True">
      <RichEditBox HorizontalAlignment="Left" Height="680" VerticalAlignment="Top" Width="1330"/>
   </c1:C1TabItem>
   ```


Copyright © GrapeCity, Inc. All rights reserved.

```
<c1:C1TabItem Header="Tasks">
    <RichEditBox HorizontalAlignment="Left" Height="680" VerticalAlignment="Top" Width="1330"/>
</c1:C1TabItem>
<c1:C1TabItem Header="Reminders">
    <RichEditBox HorizontalAlignment="Left" Height="680" VerticalAlignment="Top" Width="1330"/>
</c1:C1TabItem>
<c1:C1TabItem Header="Topics" CanUserPin="True">
    <RichEditBox HorizontalAlignment="Left" Height="680" VerticalAlignment="Top" Width="1330"/>
</c1:C1TabItem>
```

これにより、複数の **C1TabItem** が追加され、それぞれに **RichEditBox** が含まれます。2つの **C1TabItem** は、**CanUserPin** が **True** に設定され、実行時のタブの固定を許可しています。

この手順では、C1TabControl コントロールを追加してカスタマイズしました。次の手順では、プログラムを実行し、このクイックスタートで行ったことを確認します。

手順3：プロジェクトの実行

前の手順では、**C1TabControl** コントロールのプロジェクトを作成し、そのコントロールにタブページを追加し、コントロールの外観と動作を変更しました。この手順では、プログラムを実行し、C1TabControl コントロールに加えたすべての変更を確認します。

次の手順に従います。

1. [**デバッグ**]→[**デバッグ開始**]を選択して、プロジェクトを実行します。C1TabControl コントロールのタブストリップがコントロールの下端に沿って並び、端が斜めになったタブが配置されていることを確認します。
2. 最初のタブの **RichEditBox** にコンテンツを入力します。
3. 2番目のタブをクリックして、フォーカスが2番目のタブに変更されることを確認します。
4. タブの閉じるボタンをクリックして、タブが閉じられることを確認します。

おめでとうございます。これで、**TabControl for UWP** クイックスタートの4つの手順のすべてが完了しました。

TreeView for UWP

TreeView for UWP を使用すると、データ項目を階層的に表示できます。使い慣れたツリービュー UI を Windows 8 アプリケーションで使用できるようになりました。折りたたみ可能なノード、階層テンプレート、チェックボックスノード、編集、ドラッグ & ドロップ操作（ベータ版ではマウスのみ）などがサポートされています。

主な特長

TreeView 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。**TreeView for UWP** は、次の主な特長を備えています。

- **カスタマイズ可能なノード**
  ノードヘッダーはコンテンツ要素なので、任意の種類の要素をホストできます。画像、チェックボックスなど、アプリケーションに必要なあらゆる要素を追加できます。カスタマイズ可能な **EditTemplate** プロパティを使用して、編集機能を提供できます。
- **接続線の表示**
  ShowLines プロパティを設定するだけで、C1TreeView に接続線が表示されます。これは、従来の Windows ツリービューの外観を与えます。いくつかの単純なプロパティ（**LineThickness/LineStroke**）で、線の外観を調整できます。
- **階層テンプレート**

Copyright © GrapeCity, Inc. All rights reserved.

C1TreeViewItem クラスをサブクラス化しなくても、ノードタイプごとに異なるテンプレートを使用できます。

- **ノードのドラッグ &ドロップ**
  C1TreeView は、ツリー内のドラッグ &ドロップ操作をサポートします。**AllowDragDrop** プロパティを true に設定するだけで、ツリー内のノードをドラッグして並べ替えることができます。ベータ版では、キーボードとマウスでのみドラッグ &ドロップ機能がサポートされます。

クイックスタート

このクイックスタートガイドは、**TreeView 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ガイドでは、最初に Visual Studio で新しいプロジェクトを作成します。次に、アプリケーションに C1TreeView コントロールを追加し、**C1TreeView** コントロールのコンテンツ領域にコンテンツを追加します。

手順1：C1TreeView コントロールを含むアプリケーションの作成

この手順では、最初に Visual Studio で **TreeView for UWP** を使用する アプリケーションを作成します。

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、言語の下で[**Windows ストア**]を選択し、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. **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。
4. ツールボックスに移動し、[**C1TreeView**]アイコンをダブルクリックして、**MainPage.xaml** にツリービューコントロールを追加します。XAML マークアップは次のようになります。

   マークアップ

   ```
   <Page xmlns:Xaml="using:C1.Xaml" x:Class="C1TreeViewQuickStart.MainPage"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:local="using:C1TreeViewQuickStart"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d">
       <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
           <Xaml:C1TreeView />
       </Grid>
   </Page>
   ```

   C1.Xaml 名前空間と `<Xaml:C1TreeView>` タグがプロジェクトに追加されています。
5. `x:Name="Tree"` を `<Xaml:C1TreeView>` タグに追加してグリッドに名前を付けてから終了タグを追加します。次のようになります。

   マークアップ

   ```
   <Xaml:C1TreeView x:Name="Tree"></Xaml:C1TreeView>
   ```

   コントロールに一意の識別子を付けると、コードでその **C1TreeView** コントロールにアクセスできるようになります。

これで、C1TreeView コントロールを含む Windows Store アプリケーションが作成されました。次の手順では、C1TreeView コントロールの外観と動作をカスタマイズします。

手順2：C1TreeView への C1TreeView 項目の追加

このレッスンでは、C1TreeView コントロールに静的 **C1TreeView** 項目を追加する方法について説明します。XAML で **C1TreeView** コントロールに静的 C1TreeViewItem を追加するには、次の手順に従います。

Copyright © GrapeCity, Inc. All rights reserved.

1. C1TreeViewItem を追加して、"ジャンル一覧" という名前の最上位ノードを作成します。`<Xaml:C1TreeViewItem>` タグ内に `Header="ジャンル一覧"` を追加します。これにより、実行時に最上位ノードが作成されます。XAML マークアップは次のようになります。

```
マークアップ
<Xaml:C1TreeViewItem Header="ジャンル一覧"></Xaml:C1TreeViewItem>
```

2. `<Xaml:C1TreeViewItem>` タグの下に2つの子 **C1TreeViewItem** を追加して、Book List ノードの2つの子ノードを作成します。`Header="文学"` を追加します。2番目の子ノードには、`Header="ノンフィクション"` を追加します。XAML マークアップは次のようになります。

```
マークアップ
<Xaml:C1TreeViewItem Header="文学"/>
<Xaml:C1TreeViewItem Header="ノンフィクション"/>
```

3. もう1つの `<Xaml:C1TreeViewItem>` タグを追加し、2つの子ノードを含む新しい最上位ノードを作成します。XAML マークアップは次のようになります。

```
マークアップ
<Xaml:C1TreeViewItem Header="ビジネス">
<Xaml:C1TreeViewItem Header="Catch-22"/>
<Xaml:C1TreeViewItem Header="経済学"/>
```

4. 2つの `<Xaml:C1TreeViewItem>` の終了タグを追加して、**ジャンル一覧 ノードと ビジネス** ノードを閉じます。これで、MainPage.xaml には、次の XAML マークアップが含まれます。

```
マークアップ
<Page xmlns:Xaml="using:C1.Xaml" x:Class="C1TreeViewQuickStart.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" xmlns:local="using:C1TreeViewQuickStart"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006" mc:Ignorable="d">
   <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
<Xaml:C1TreeView x:Name="Tree">
      <Xaml:C1TreeViewItem Header="ジャンル一覧">
         <Xaml:C1TreeViewItem Header="文学"/>
         <Xaml:C1TreeViewItem Header="ノンフィクション"/>
         <Xaml:C1TreeViewItem Header="ビジネス">
            <Xaml:C1TreeViewItem Header="Catch-22"/>
            <Xaml:C1TreeViewItem Header="経済学"/>
         </Xaml:C1TreeViewItem>
      </Xaml:C1TreeViewItem>
</Xaml:C1TreeView>
   </Grid>
</Page>
```

5. プロジェクトを実行します。Book ノードは展開されていません。このノードは、矢印マークをクリックすることで展開できます。

この手順では、C1TreeView コントロールにいくつかの **C1TreeView** 項目を追加しました。次の手順では、**C1TreeView** コントロールの動作と外観をカスタマイズします。

手順3：TreeView の外観と動作のカスタマイズ

前の手順では、Visual Studio で、XAML マークアップを使用して C1TreeViewItem を作成しました。この手順では、XAML コードを使用して **C1TreeView** コントロールの外観と動作をカスタマイズします。

Copyright © GrapeCity, Inc. All rights reserved.

TreeView for UWP をカスタマイズするには、次の手順に従います。

1. `<Xaml:C1TreeView>` タグ内にカーソルを置きます。`<Xaml:C1TreeView>` タグ内に `SelectionMode="Extended"` を追加します。これにより、[Shift]キーまたは[Ctrl]キーを押しながら複数のツリー項目を選択できる最上位ノードが作成されます。XAML マークアップは次のようになります。

マークアップ
<Xaml:C1TreeView x:Name="Tree" SelectionMode="Extended">

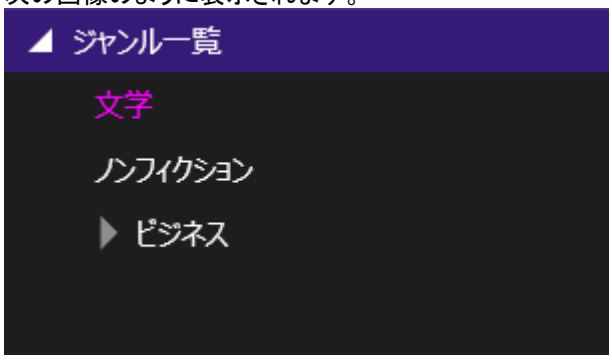
2. 最初の `<Xaml:C1TreeViewItem>` タグ内にカーソルを置きます。`<Xaml:C1TreeViewItem>` 内に `IsExpanded="True" IsSelected="True"` を追加します。これにより、実行時に選択および展開されて表示される最上位ノードが作成されます。XAML マークアップは次のようになります。

マークアップ
<Xaml:C1TreeViewItem Header="ジャンル一覧" IsExpanded="True" IsSelected="True">

3. `<Xaml:C1TreeViewItem Header="文学">` タグに移動します。`<Xaml:C1TreeViewItem Header="文学">` タグ内に `Foreground="Fuchsia"` を追加します。"ビジネス" ツリー項目のテキストが赤紫色で表示されます。XAML マークアップは次のようになります。

マークアップ
<Xaml:C1TreeViewItem Header="文学" Foreground="Fuchsia"/>

4. [**デバッグ**]メニューから[**デバッグ開始**]を選択し、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。次の画像のように表示されます。



C1TreeView の動作と外観が次のように変更されていることを確認します。
   - **C1TreeView** が展開されて表示されます。
   - 最初の **C1TreeViewItem** が選択されて表示されます。
   - 2番目の **C1TreeViewItem** のテキストが赤紫色で表示されます。
   - 項目を選択する際に[CTRL]キーまたは[SHIFT]キーを押しながら複数の **C1TreeViewItem** を選択できます。

おめでとうございます。これで、**TreeView for UWP** クイックスタートは終了です。このクイックスタートでは、**TreeView for UWP** アプリケーションを作成してカスタマイズし、静的 C1TreeViewItem を追加し、コントロールの実行時の動作をいくつか確認しました。

C1TreeView 構造

C1TreeView クラスは、次の2つの要素から成る1つの **StackPanel** です。

- 実際のノードを表すヘッダー。子を展開/折りたたむためのボタンを1つ持ちます。
- 別の **StackPanel** から成る本体。他のノードを格納します。

あるノードに画像を追加するには、その最初の子(ヘッダー)を取得し、それを **StackPanel** にキャストし、画像要素を任意の場所に挿入します。次に例を示します。

Visual Basic コードの書き方

Visual Basic

Copyright © GrapeCity, Inc. All rights reserved.

```
Dim nodeHeader As StackPanel = TryCast(TreeNode.Children(0), StackPanel)
nodeHeader.Children.Insert(0, myImage)
```

C# コードの書き方

C#

```
StackPanel nodeHeader = TreeNode.Children[0] as StackPanel;
nodeHeader.Children.Insert(0, myImage);
```

TreeView の作成

C1TreeView コントロールに追加する C1TreeViewItem は、XAML マークアップまたはコードビハインドで静的な項目として定義できます。または、ページまたはユーザーコントロール上に次のいずれかの方法を使用して定義することもできます。

- XAML 構文を使用するか、コードビハインドファイルを使用したプログラムによる静的な作成。
- コンストラクタを使用して **C1TreeViewItem** クラスの新しいインスタンスを作成する動的な作成。
- C1TreeView を **SiteMapDataSource**、**XMLDataSource**、または **AccessDataSource** に連結することによるデータソースの作成。

静的な TreeView の作成

ツリー内の各ノードは、ツリーノードのテキストプロパティと値プロパティで定義される名前/値のペアで表されます。ノードのテキストはレンダリングされますが、ノードの値はレンダリングされず、通常はポストバックイベントを処理するための追加データとして使用されます。

静的なメニューは、ツリービュー構造を作成するための最も簡単な方法です。

XAML 構文を使用して静的 C1TreeViewItem を表示するには、最初に、C1TreeView コントロールの開始タグと終了タグの間に `<Xaml:C1TreeViewItem>` の開始タグと終了タグをネストします。次に、`<Xaml:C1TreeViewItem>` の開始タグと終了タグの間に `<Xaml:C1TreeViewItem>` 要素をネストして、ツリービュー構造を作成します。各 `<Xaml:C1TreeViewItem>` 要素は、コントロール内の1つのノードを表し、**C1TreeViewItem** オブジェクトにマップされます。

宣言構文を使用すると、C1TreeViewItem をページ内でインライン定義できます。

次に例を示します。

マークアップ

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
  <Xaml:C1TreeView x:Name="Tree">
    <Xaml:C1TreeViewItem Header="ジャンル一覧" IsExpanded="True" IsSelected="True">
      <Xaml:C1TreeViewItem Header="文学"/>
      <Xaml:C1TreeViewItem Header="ノンフィクション"/>
      <Xaml:C1TreeViewItem Header="ビジネス">
        <Xaml:C1TreeViewItem Header="Catch-22"/>
        <Xaml:C1TreeViewItem Header="経済学"/>
      </Xaml:C1TreeViewItem>
    </Xaml:C1TreeViewItem>
  </Xaml:C1TreeView>
</Grid>
```

動的な TreeView の作成

Copyright © GrapeCity, Inc. All rights reserved.

動的なツリービューは、サーバー側またはクライアント側で作成できます。サーバー側で動的なツリービューを作成する場合は、**C1TreeView** クラスの新しいインスタンスを動的に作成するためのコンストラクタを使用します。次に例を示します。

Visual Basic コードの書き方

Visual Basic

```
Namespace TreeViewQuickStart
    Public Partial Class MainPage
        Inherits UserControl
        Public Sub New()
            InitializeComponent()

            InitializeTreeView()
        End Sub
        Private Sub InitializeTreeView()

            ' 設計時に追加された項目を削除します

            Tree.Items.Clear()

            Dim booklist As New C1TreeViewItem()
            booklist.Header = "Book List"
            Tree.Items.Add(booklist)

            ' 子項目を追加します
            Dim language As New C1TreeViewItem()
            language.Header = "Language Books"
            booklist.Items.Add(language)

            ' 子項目を追加します
            Dim security As New C1TreeViewItem()
            security.Header = "Security Books"
            booklist.Items.Add(security)

            ' 子項目を追加します
            Dim classic As New C1TreeViewItem()
            classic.Header = "Classic Books"
            booklist.Items.Add(classic)
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        ' 子項目を追加します
        Dim subclassic As New C1TreeViewItem()
        subclassic.Header = "Catch-22"
        classic.Items.Add(subclassic)
        Dim subclassic2 As New C1TreeViewItem()
        subclassic2.Header = "The Great Gatsby"
        classic.Items.Add(subclassic2)
    End Sub
  End Class
End Namespace
```

C# コードの書き方

C#

```
 namespace TreeViewQuickStart
{
   public partial class MainPage : UserControl
   {
      public MainPage()
      {
         InitializeComponent();
         InitializeTreeView();

      }
     void InitializeTreeView()
     {

         // 設計時に追加された項目を削除します

         Tree.Items.Clear();

         C1TreeViewItem booklist = new C1TreeViewItem();
         booklist.Header = "Book List";
        Tree.Items.Add(booklist);

        // 子項目を追加します
       C1TreeViewItem language = new C1TreeViewItem();
       language.Header = "Language Books";
       booklist.Items.Add( language );
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        // 子項目を追加します
        C1TreeViewItem security = new C1TreeViewItem();
        security.Header = "Security Books";
        booklist.Items.Add(security);

       // 子項目を追加します
       C1TreeViewItem classic = new C1TreeViewItem();
       classic.Header = "Classic Books";
       booklist.Items.Add(classic);

      // 子項目を追加します
      C1TreeViewItem subclassic = new C1TreeViewItem();
      subclassic.Header = "Catch-22";
      classic.Items.Add(subclassic);
      C1TreeViewItem subclassic2 = new C1TreeViewItem();
      subclassic2.Header = "The Great Gatsby";
      classic.Items.Add(subclassic2);
}
   }
}
```

データソースを活用した TreeView 項目の作成

XMLDataSource、**SiteMapDataSource** などの階層化データソースコントロールから TreeView 項目を作成できます。これにより、コードを編集しなくてもツリービュー項目を更新できるようになります。

C1TreeView の **ItemsSource** として複数レベルのデータを使用する場合は、それらの項目に対して1つの C1HierarchicalDataTemplate を指定する必要があります。

このテンプレートは、**C1TreeView** に次のレベルのデータがある場所を通知します。これは、**C1HierarchicalDataTemplate** の ItemsSource プロパティによって行われます。

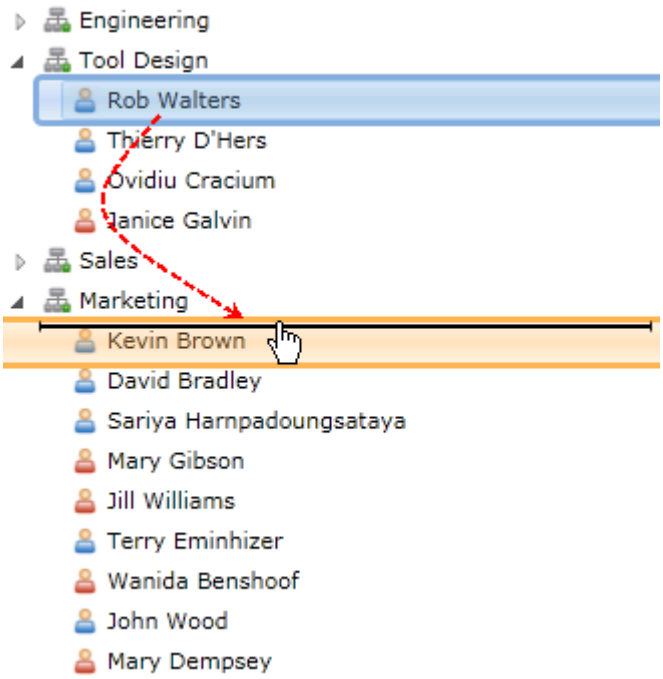
TreeView の動作

ノードのドラッグ &ドロップ

AllowDragDrop プロパティが **True** に設定されている場合は、**C1TreeViewItem** をノード、ノードの間、または1つのツリーから別のツリーにドラッグ &ドロップすることができます。

次の図では、**C1TreeViewItem** を1つの **C1TreeView** から別の **C1TreeView** にドラッグしています。**DragDropArrowMarker** プロパティまたは DragDropLineMarker プロパティのいずれかを適用すると、矢印または垂直線を使用して、**C1TreeViewItem** のドロップ先を視覚的に示すことができます。

Copyright © GrapeCity, Inc. All rights reserved.



オンデマンドの読み込み

アプリケーションの起動時に各ノードをすべて生成するのではなく、ユーザーがノードを展開すると、オンデマンドでノードにデータが挿入される遅延ロードという技術を使用することができます。これにより、アプリケーションの読み込み速度が向上し、リソースを効率よく使用できます。

ノードの遅延ロードを実装するには、次のコードを使用します。

Visual Basic コードの書き方

Visual Basic

```
Public Sub New()
InitializeComponent()
' ここは変更しません
' ...
' C1TreeView を初期化します
InitializeTreeView()
End Sub
Private Sub InitializeTreeView()
' 設計時に追加された項目を削除します
_tv.Items.Clear()
' アセンブリ内のすべての型をスキャンします
For Each t As Type In _tv.[GetType]().Assembly.GetTypes()
    If t.IsPublic AndAlso Not t.IsSpecialName AndAlso Not t.IsAbstract Then
        ' この型のノードを追加します
        Dim node As New C1TreeViewItem()
        node.Header = t.Name
        node.FontWeight = FontWeights.Bold
        _tv.Items.Add(node)
        ' プロパティ、イベント、およびメソッドのサブノードを追加します
        node.Items.Add(CreateMemberNode("Properties", t, MemberTypes.[Property]))
        node.Items.Add(CreateMemberNode("Events", t, MemberTypes.[Event]))
        node.Items.Add(CreateMemberNode("Methods", t, MemberTypes.Method))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        End If
Next
End Sub
Private Function CreateMemberNode(header As String, memberTypes As MemberTypes) As C1TreeViewItem
' ノードを作成します
Dim node As New C1TreeViewItem()
node.Header = header
node.Foreground = New SolidColorBrush(Colors.DarkGray)
' ノードを展開する前にノードにデータを挿入するためのイベントハンドラを登録します
AddHandler node.Expanding, AddressOf node_Expanding
' ノードにデータを挿入するために必要な情報を保存します
node.Tag = memberTypes
' このノードを展開できるように、ダミーノードを追加します
node.Items.Add(New C1TreeViewItem())
node.IsExpanded = False
' 終了します
Return node
End Function
'ノードにデータを挿入します
Private Sub node_Expanding(sender As Object, e As RoutedEventArgs)
' イベントが発生したノードを取得します
Dim node As C1TreeViewItem = TryCast(sender, C1TreeViewItem)
' イベントハンドラの登録を解除します（ノードにデータを挿入したら、このハンドラは不要になります）
RemoveHandler node.Expanding, AddressOf node_Expanding
' ダミーノードを削除します
node.Items.Clear()
' ノードにデータを挿入します
Dim type As Type = DirectCast(node.Parent.Tag, Type)
Dim memberTypes As MemberTypes = DirectCast(node.Tag, MemberTypes)
Dim bf As BindingFlags = BindingFlags.[Public] Or BindingFlags.Instance
For Each mi As MemberInfo In type.GetMembers(bf)
        If mi.MemberType = memberTypes Then
            If Not mi.Name.StartsWith("get_") AndAlso Not mi.Name.StartsWith("set_") Then
                Dim item As New C1TreeViewItem()
                item.Header = mi.Name
                item.FontSize = 12
                node.Items.Add(item)
            End If
        End If
Next
End Sub
```

C# コードの書き方

C#

```
public Page()
{
   InitializeComponent();
   // ここは変更しません
   // ...
   // C1TreeView を初期化します
   InitializeTreeView();
```


Copyright © GrapeCity, Inc. All rights reserved.

```
}
void InitializeTreeView()
{
 // 設計時に追加された項目を削除します
 _tv.Items.Clear();
  // アセンブリ内のすべての型をスキャンします
 foreach (Type t in _tv.GetType().Assembly.GetTypes())
 {
  if (t.IsPublic && !t.IsSpecialName && !t.IsAbstract)
  {
   // この型のノードを追加します
   C1TreeViewItem node = new C1TreeViewItem();
   node.Header = t.Name;
   node.FontWeight = FontWeights.Bold;
   _tv.Items.Add(node);
   // プロパティ、イベント、およびメソッドのサブノードを追加します
   node.Items.Add(CreateMemberNode("Properties", t, MemberTypes.Property));
   node.Items.Add(CreateMemberNode("Events", t, MemberTypes.Event));
   node.Items.Add(CreateMemberNode("Methods", t, MemberTypes.Method));
  }
 }
}
C1TreeViewItem CreateMemberNode(string header, MemberTypes memberTypes)
{
 // ノードを作成します
 C1TreeViewItem node = new C1TreeViewItem();
 node.Header = header;
 node.Foreground = new SolidColorBrush(Colors.DarkGray);
  // ノードを展開する前にノードにデータを挿入するためのイベントハンドラを登録します
 node.Expanding += node_Expanding;
  // ノードにデータを挿入するために必要な情報を保存します
 node.Tag = memberTypes;
  // このノードを展開できるように、ダミーノードを追加します
 node.Items.Add(new C1TreeViewItem());
 node.IsExpanded = false;
  // 終了します
 return node;
}
//ノードにデータを挿入します
void node_Expanding(object sender, RoutedEventArgs e)
{
 // イベントが発生したノードを取得します
 C1TreeViewItem node = sender as C1TreeViewItem;
 // イベントハンドラの登録を解除します（ノードにデータを挿入したら、このハンドラは不要になります）
 node.Expanding -= node_Expanding;
  // ダミーノードを削除します
 node.Items.Clear();
  // ノードにデータを挿入します
 Type type = (Type)node.Parent.Tag;
 MemberTypes memberTypes = (MemberTypes)node.Tag;
  BindingFlags bf = BindingFlags.Public | BindingFlags.Instance;
 foreach (MemberInfo mi in type.GetMembers(bf))
```


Copyright © GrapeCity, Inc. All rights reserved.

```
 {
  if (mi.MemberType == memberTypes)
  {
   if (!mi.Name.StartsWith("get_") && !mi.Name.StartsWith("set_"))
   {
    C1TreeViewItem item = new C1TreeViewItem();
    item.Header = mi.Name;
    item.FontSize = 12;
    node.Items.Add(item);
   }
  }
 }
}
```

この実装は、ユーザーがノードを展開しようとしたときに、ノードにデータを挿入できるように、**Expanding** イベントのイベントハンドラを登録します。**Tag** プロパティには、ノードにデータを挿入するために必要な情報を保存します。最後に、ユーザーがこのノードを展開して、ノードにデータを挿入する Expanding イベントをトリガできるように、ダミーの子ノードを追加します。

Tag プロパティを使用する代わりに、C1TreeViewItem からカスタムクラスを継承し、そのクラスにすべての遅延ロードロジックを組み込むこともできるはずです。その方が洗練された方法ですが、ではテンプレートの継承がサポートされていません。テンプレートを持つクラス（**Button**、**C1TreeViewItem** など）からクラスを継承しても、テンプレートは継承されないため、テンプレートを各自で提供する必要があります。そうしないと、その派生クラスは単に空のコントロールになります。

ノードの選択

実行時にノードをクリックすると、そのノードは自動的に選択中としてマークされます。カスタム機能を提供するために、ノードをクリックすると、SelectionChanged イベントが発生します。ノードがクリックされなくても選択中としてマークするには、IsSelected プロパティを有効にします。

ユーザーが新しい項目を選択すると、C1TreeView は SelectionChanged イベントを発生します。そこで、SelectedItem プロパティを使用すると、選択中の項目を取得できます。

それには、いくつかの方法があります。1つの方法は、項目の作成時に、各 **C1TreeViewItem** の Tag プロパティに追加データを割り当てます。後で、その Tag プロパティを調べることで、その情報を取得できます。次に例を示します。

Visual Basic コードの書き方

Visual Basic

```
' ノードを作成し、その Tag プロパティにデータを割り当てます
Dim item As New C1TreeViewItem()
item.Header = "Beverages"
item.Tag = beveragesID
```

C# コードの書き方

C#

```
// ノードを作成し、その Tag プロパティにデータを割り当てます
C1TreeViewItem item = new C1TreeViewItem();
item.Header = "Beverages";
item.Tag = beveragesID;
```

後で、この情報を適切な場面で使用します。

Visual Basic コードの書き方

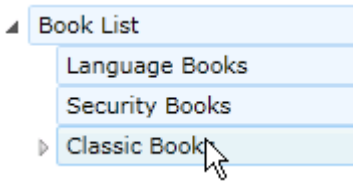
Copyright © GrapeCity, Inc. All rights reserved.

```
Visual Basic
Dim item As C1TreeViewItem = _tv.SelectedItem
    ' Beverages ノードを処理します
If TypeOf item.Tag Is Integer AndAlso CInt(item.Tag) = beveragesID Then
End If
```

C# コードの書き方

```
C#
C1TreeViewItem item = _tv.SelectedItem;
if (item.Tag is int && (int)item.Tag == beveragesID)
{
    // Beverages ノードを処理します
}
```

SelectionMode プロパティが **Multiple** に設定されている場合は、[Ctrl]キーを押しながら複数のノードをクリックすることで、一度に複数のノードを選択できます。ノードを選択解除するには、そのノードを再度クリックします。次の **C1TreeView**では、それらのノードが選択中としてマークされています。



ノードの移動

C1TreeView は、マウスとキーボードによる移動をサポートします。

マウスを使用した C1TreeViewItem 間の移動

次の表は、**C1TreeViewItem** 間を移動するアクションに対応するマウスコマンドを示します。

アクション	マウスコマンド
ノードの展開	ノード名の左側にあるプラス記号をクリックします。
ノードの折りたたみ	ノード名の左側にあるマイナス記号をクリックします。
ノードの選択	ノード名をクリックします。

キーボードを使用した C1TreeViewItem 間の移動

次の表は、**C1TreeViewItem** 間を移動するアクションに関連するキーを示します。

アクション	キーボードコマンド
ノードの展開	[+]キー
ノードの折りたたみ	[-]キー
ノードを上に移動	[↑]キー
ノードを下に移動	[↓]キー

Copyright © GrapeCity, Inc. All rights reserved.

複数ノードの選択	マウス +[Ctrl]キー + クリック

レイアウトおよび外観

以下のトピックでは、**C1TreeView** 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。組み込みのレイアウトオプションを使用して、グリッドやキャンバスなどのコントロールをパネル内でレイアウトできます。テーマを使用することで、ツリービューの外観をカスタマイズしたり、の XAML ベースのスタイル設定を活用することができます。

Header プロパティを使用して、ツリービュー項目の外観をカスタマイズできます。**Header** は、画像とテキストを含むスタックパネルなど、任意のオブジェクトに設定できます。

マークアップ

```
<C1TreeViewItem>
  <Xaml:C1TreeViewItem.Header>
    <StackPanel Orientation="Horizontal">
      <Image Source="myImage.jpg"/>
      <TextBlock Text="私のテキスト"/>
    </StackPanel>
  </Xaml:C1TreeViewItem.Header>
</Xaml:C1TreeViewItem>
```

外観プロパティ

TreeView for UWP には、コントロールの外観をカスタマイズするためのいくつかのプロパティがあります。コントロールに表示されるテキストの外観を変更したり、コントロールのグラフィック要素をカスタマイズすることができます。以下のトピックでは、これらの外観プロパティの一部について説明します。

テキストのプロパティ

以下のプロパティを使用すると、C1TreeView コントロールのテキストの外観をカスタマイズできます。

プロパティ	説明
FontFamily	コントロールのフォントファミリを取得または設定します。これは依存プロパティです。
FontSize	フォントサイズを取得または設定します。これは依存プロパティです。
FontStretch	フォントを画面上で伸縮する比率を取得または設定します。これは依存プロパティです。
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。
FontWeight	指定されたフォント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。
C1HierarchicalPresenter.Header	コントロールにラベルを付ける項目を取得または設定します。

コンテンツ配置のプロパティ

次のプロパティを使用して、C1TreeView コントロールのヘッダーおよびコンテンツ領域のコンテンツの配置をカスタマイズできます。

プロパティ	説明

Copyright © GrapeCity, Inc. All rights reserved.

Padding	コントロールの内側のパディングを取得または設定します。これは依存プロパティです。
HorizontalAlignment	パネルや項目コントロールなどの親要素内に置かれる要素に適用される水平配置の特性を取得または設定します。これは依存プロパティです。
HorizontalContentAlignment	コントロールのコンテンツの水平方向の配置を取得または設定します。これは依存プロパティです。
VerticalAlignment	パネルや項目コントロールなどの親要素内に置かれる要素に適用される垂直配置の特性を取得または設定します。これは依存プロパティです。
VerticalContentAlignment	コントロールのコンテンツの垂直方向の配置を取得または設定します。これは依存プロパティです。

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して、C1TreeView 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。

C1TreeView のテンプレート

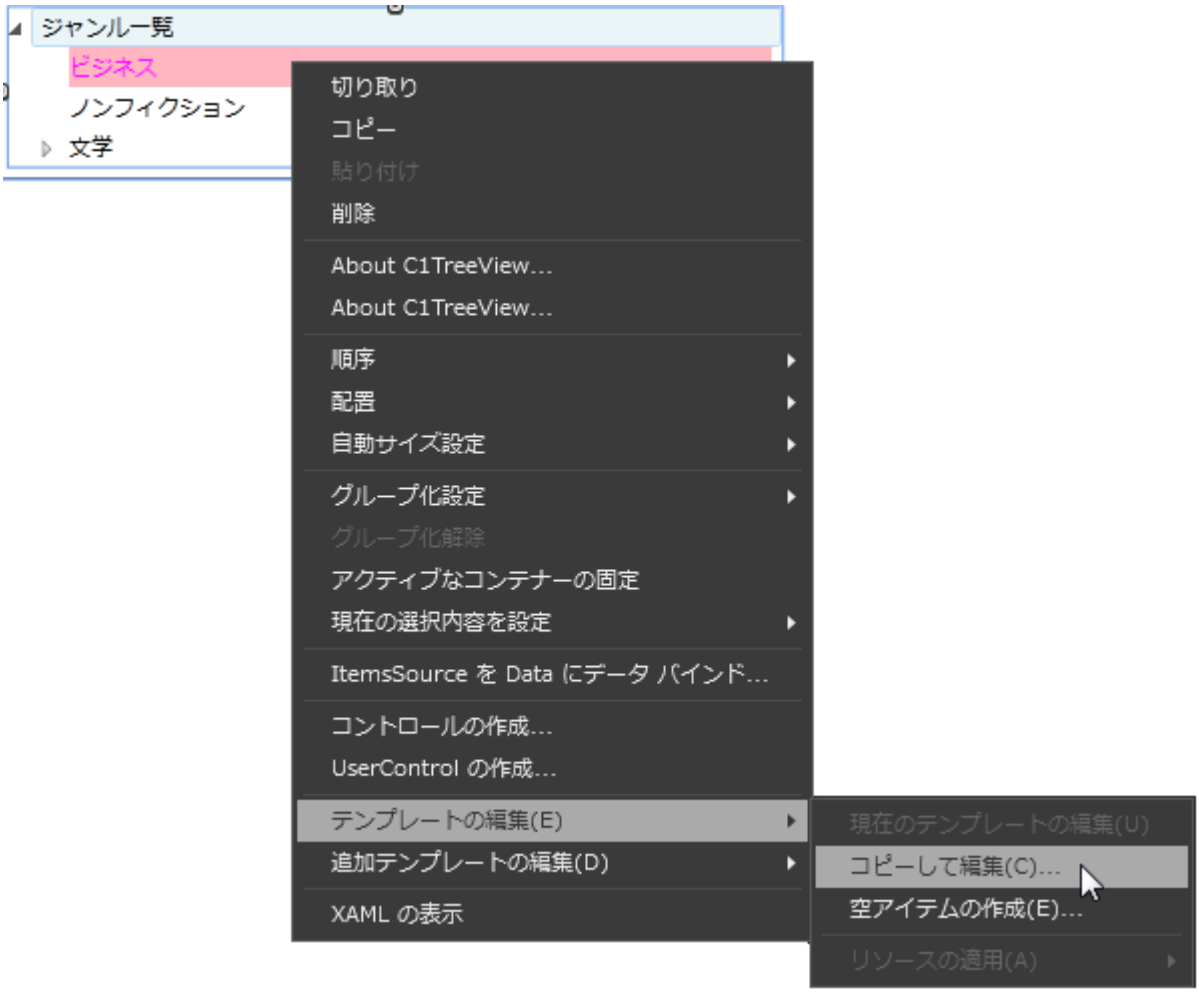
コントロールを使用する主なメリットの１つは、これが自由にカスタマイズできるユーザーインタフェースを持つ「外観のない」コ

Copyright © GrapeCity, Inc. All rights reserved.

ントロールだからです。アプリケーションで独自のユーザーインタフェース（UI）、つまり「外観」を設計するのと同様に、**TreeView for UWP** によって管理されるデータにも独自の UI を提供できます。Extensible Application Markup Language（XAML。「ザムル」と発音する）は、コードを記述することなく独自の UI を設計するための簡単な方法を提供する XML ベースの宣言型言語です。

テンプレートへのアクセス

テンプレートにアクセスするには、Microsoft Expression Blend で、C1TreeView コントロールを選択し、メニューから[**テンプレートの編集**]を選択します。[**コピーの編集**]を選択して現在のテンプレートのコピーを作成して編集するか、[**空のテンプレートの作成**]を選択して新しい空のテンプレートを作成します。



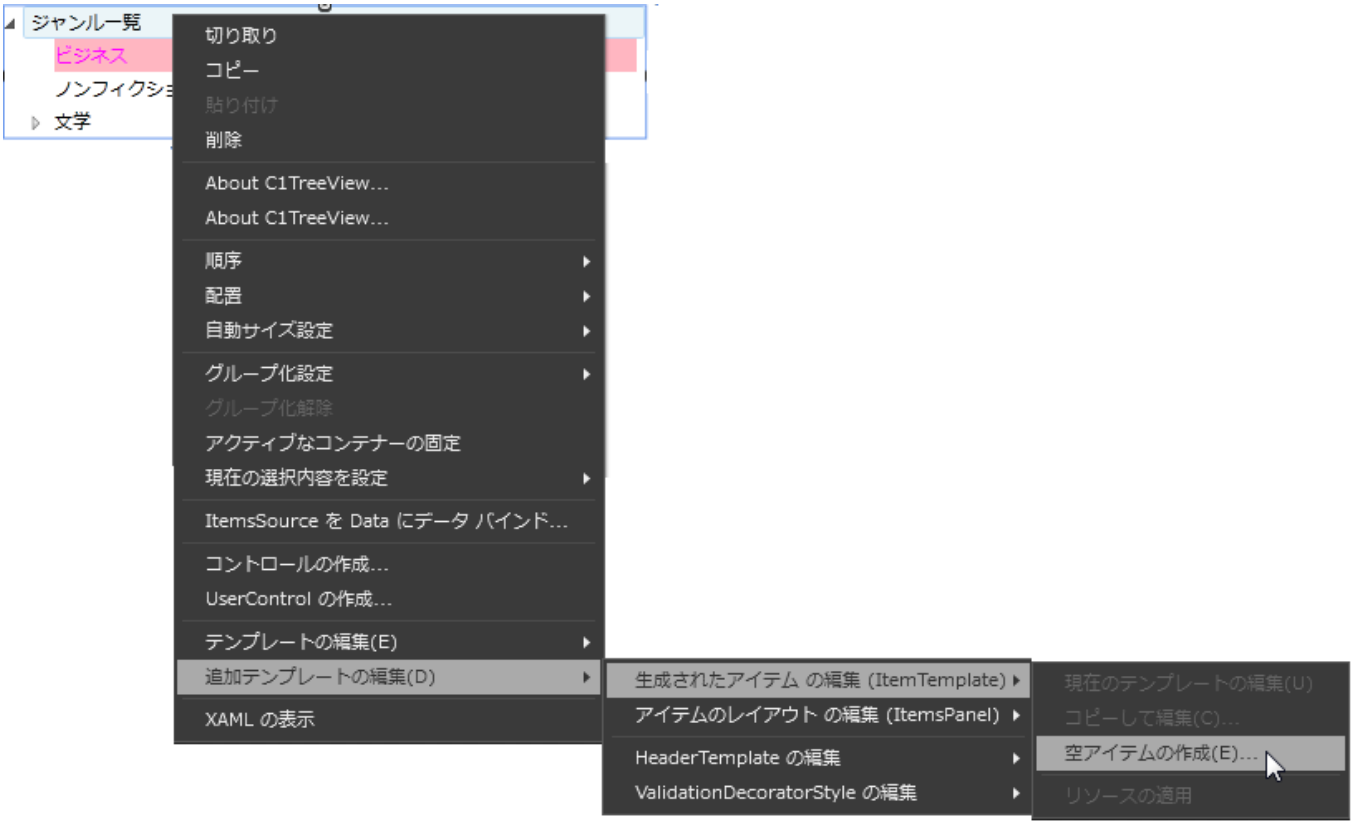
新しく作成されたテンプレートは、[**オブジェクトとタイムライン**]ウィンドウに表示されます。Template プロパティを使用して、テンプレートをカスタマイズできます。

> **メモ:** メニューから新しいテンプレートを作成する場合、テンプレートはそのテンプレートのプロパティに自動的にリンクされます。手作業でテンプレートの XAML を作成する場合は、作成したテンプレートに適切なテンプレートプロパティをリンクする必要があります。

追加のテンプレート

デフォルトテンプレートのほかに、C1TreeView コントロールには追加のテンプレートがいくつかあります。これらの追加テンプレートには、Microsoft Expression Blend からもアクセスできます。Blend で C1TreeView コントロールを選択し、メニューから[**追加テンプレートの編集**]を選択します。テンプレートを選択し、[**空のテンプレートの作成**]を選択します。

Copyright © GrapeCity, Inc. All rights reserved.



C1TreeView のスタイル

TreeView for UWP の C1TreeView 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
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。

ClearStyle

TreeView for UWP は、コントロールのテンプレートを変更することなくコントロールの色を簡単に変更できる ComponentOne の新しい ClearStyle 技術をサポートします。色のプロパティをいくつか設定するだけで、C1TreeView コントロールのスタイルを簡単に設定できます。次の表に、**C1TreeView** でサポートされているプロパティを一覧します。

プロパティ	説明
Background	**C1DockControl** の塗りつぶしに使用される背景を取得または設定します。
MouseOverBrush	マウスポインタが置かれたコントロールを強調表示するために使用されるブラシを取得または設定します。
SelectedBackground	選択された項目の背景色を取得または設定します。

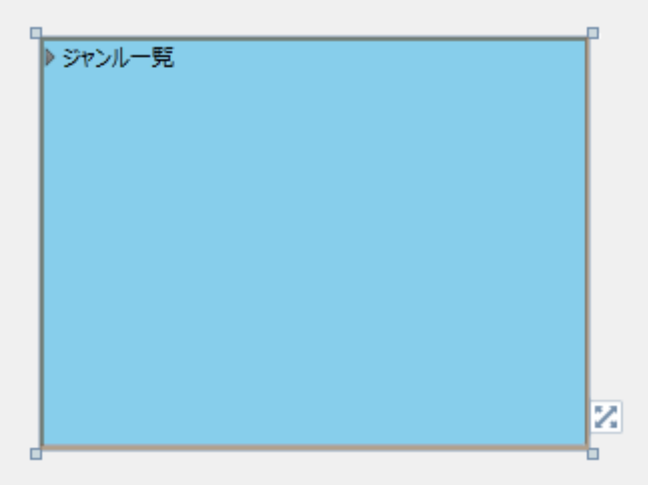
これらのプロパティを設定することで、**C1TreeView** の外観を完全に変更できます。たとえば、**C1TreeView.Background** プロパティを **SkyBlue** に設定する場合の XAML マークアップは次のようになります。

マークアップ

Copyright © GrapeCity, Inc. All rights reserved.

```
<Xaml:C1TreeView Background="SkyBlue" x:Name="Tree">
        <Xaml:C1TreeViewItem Header="ジャンル一覧">
          <Xaml:C1TreeViewItem Header="文学"/>
          <Xaml:C1TreeViewItem Header="ノンフィクション"/>
          <Xaml:C1TreeViewItem Header="ビジネス">
            <Xaml:C1TreeViewItem Header="経済学"/>
            <Xaml:C1TreeViewItem Header="マーケティング"/>
          </Xaml:C1TreeViewItem>
        </Xaml:C1TreeViewItem>
    </Xaml:C1TreeView>
```

C1TreeView は次のようになります。

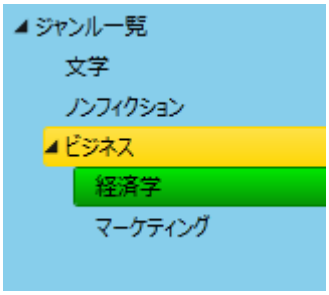


その他のプロパティも試して、**C1TreeView** 要素の外観をすばやく変更してみてください。たとえば、次の XAML は、**Background**、**MouseOverBrush**、**SelectedBackground** の各プロパティを設定します。

マークアップ

```
 <Xaml:C1TreeView Background="SkyBlue" MouseOverBrush="Gold" SelectedBackground="Green"
x:Name="Tree">
        <Xaml:C1TreeViewItem Header="ジャンル一覧">
          <Xaml:C1TreeViewItem Header="文学"/>
          <Xaml:C1TreeViewItem Header="ノンフィクション"/>
          <Xaml:C1TreeViewItem Header="ビジネス">
            <Xaml:C1TreeViewItem Header="経済学"/>
            <Xaml:C1TreeViewItem Header="マーケティング"/>
          </Xaml:C1TreeViewItem>
        </Xaml:C1TreeViewItem>
    </Xaml:C1TreeView>
```

プロジェクトを実行すると、C1TreeView は次のようになります。



Copyright © GrapeCity, Inc. All rights reserved.

ヘッダー内にマウスポインタを置くと、ヘッダーは黄色になります。項目を選択すると、項目は緑色で表示されます。

タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio .NET でのプログラミングに精通しており、**C1TreeView** 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。**TreeView 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、**TreeView 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。

また、タスク別ヘルプトピックは、新しい Windows ストアプロジェクトが既に作成されていることを前提としています。

コードを使用した C1TreeViewItem の追加

コードビハインドファイルで **C1TreeView** コントロールに静的 **C1TreeView** 項目を追加するには、次のようにアプリケーションのコードを編集します。

Visual Basic コードの書き方

Visual Basic

```
Imports C1.Xaml
Class MainPage
    Public Sub New()
        InitializeComponent()
        InitializeTreeView()
    End Sub
    Private Sub InitializeTreeView()
        ' 設計時に追加された項目を削除します
        Tree.Items.Clear()
        Dim booklist As New C1TreeViewItem()
        booklist.Header = "ジャンルー覧"
        Tree.Items.Add(booklist)
        ' 子項目を追加します
        Dim language As New C1TreeViewItem()
        language.Header = "文学"
        booklist.Items.Add(language)
        ' 子項目を追加します
        Dim security As New C1TreeViewItem()
        security.Header = "ノンフィクション"
        booklist.Items.Add(security)
        ' 子項目を追加します
        Dim classic As New C1TreeViewItem()
        classic.Header = "ビジネス"
        booklist.Items.Add(classic)
        ' 子項目を追加します
        Dim subclassic As New C1TreeViewItem()
        subclassic.Header = "Catch-22"
        classic.Items.Add(subclassic)
        Dim subclassic2 As New C1TreeViewItem()
        subclassic2.Header = "経済学"
        classic.Items.Add(subclassic2)
    End Sub
End Class
```


Copyright © GrapeCity, Inc. All rights reserved.

C# コードの書き方

C#

```
using C1.Xaml;
public MainPage()
{
    InitializeComponent();
    InitializeTreeView();
}
void InitializeTreeView()
{
    // 設計時に追加された項目を削除します
    Tree.Items.Clear();
    C1TreeViewItem booklist = new C1TreeViewItem();
    booklist.Header = "ジャンルー覧";
    Tree.Items.Add(booklist);
   // 子項目を追加します
    C1TreeViewItem language = new C1TreeViewItem();
    language.Header = "文学";
    booklist.Items.Add( language );
    // 子項目を追加します
    C1TreeViewItem security = new C1TreeViewItem();
    security.Header = "ノンフィクション";
    booklist.Items.Add(security);
    // 子項目を追加します
    C1TreeViewItem classic = new C1TreeViewItem();
    classic.Header = "ビジネス";
    booklist.Items.Add(classic);
    // 子項目を追加します
    C1TreeViewItem subclassic = new C1TreeViewItem();
    subclassic.Header = "Catch-22";
    classic.Items.Add(subclassic);
    C1TreeViewItem subclassic2 = new C1TreeViewItem();
    subclassic2.Header = "経済学";
    classic.Items.Add(subclassic2);
}
```

TreeView の SelectedItem のテキストまたは値の取得

Header プロパティは、**C1TreeViewItem** に含まれる値を返します。次のコードを使用して、その文字列値を取得できます。

Visual Basic コードの書き方

Visual Basic

```
Dim item As C1TreeViewItem = _tree.SelectedItem
_textBlock.Text = item.Header.ToString()
```

C# コードの書き方

C#

Copyright © GrapeCity, Inc. All rights reserved.

```
C1TreeViewItem item = _tree.SelectedItem;
_textBlock.Text = item.Header.ToString();
```

ここまでの成果

アプリケーションを実行し、**Header** プロパティが **C1TreeViewItem** に含まれる値を返すことを確認します。

TreeView へのチェックボックスの追加

C1TreeView コントロールにチェックボックスを簡単に追加できます。チェックボックスはテキストの左側に表示され、これを使用して、ユーザーがツリービュー項目を選択できるようになります。次の XAML マークアップは、C1TreeView にチェックボックスを追加します。

XAML

マークアップ

```
<Xaml:C1TreeView Name="C1TreeView1" Height="300" Width="200" >
  <Xaml:C1TreeViewItem IsExpanded="True" Margin="10">
    <Xaml:C1TreeViewItem.Header>
      <CheckBox>
        <CheckBox.Content>
          <TextBlock Text="デスクトップ" />
        </CheckBox.Content>
      </CheckBox>
    </Xaml:C1TreeViewItem.Header>
  <Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem.Header>
      <CheckBox>
        <CheckBox.Content>
          <TextBlock Text="ユーザー" />
        </CheckBox.Content>
      </CheckBox>
    </Xaml:C1TreeViewItem.Header>
  </Xaml:C1TreeViewItem>
  <Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem.Header>
      <CheckBox>
        <CheckBox.Content>
          <TextBlock Text="パブリック" />
        </CheckBox.Content>
      </CheckBox>
    </Xaml:C1TreeViewItem.Header>
    <Xaml:C1TreeViewItem>
      <Xaml:C1TreeViewItem.Header>
        <CheckBox>
          <CheckBox.Content>
          <TextBlock Text="お気に入り" />
          </CheckBox.Content>
        </CheckBox>
      </Xaml:C1TreeViewItem.Header>
  </Xaml:C1TreeViewItem>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
    <Xaml:C1TreeViewItem>
      <Xaml:C1TreeViewItem.Header>
        <CheckBox>
          <CheckBox.Content>
          <TextBlock Text="パブリック ダウンロード" />
          </CheckBox.Content>
        </CheckBox>
      </Xaml:C1TreeViewItem.Header>
</Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem>
      <Xaml:C1TreeViewItem.Header><
        <CheckBox><
          <CheckBox.Content>
          <TextBlock Text="パブリック ミュージック" />
          </CheckBox.Content>
        </CheckBox>
      </Xaml:C1TreeViewItem.Header>
</Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem>
      <Xaml:C1TreeViewItem.Header>
        <CheckBox>
          <CheckBox.Content>
          <TextBlock Text="パブリック ピクチャ" />
          </CheckBox.Content>
        </CheckBox>
      </Xaml:C1TreeViewItem.Header>
</Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem>
      <Xaml:C1TreeViewItem.Header>
        <CheckBox>
          <CheckBox.Content>
          <TextBlock Text="パブリック ビデオ" />
          </CheckBox.Content>
        </CheckBox>
      </Xaml:C1TreeViewItem.Header>
</Xaml:C1TreeViewItem>
      </Xaml:C1TreeViewItem>
      <Xaml:C1TreeViewItem IsExpanded="True">
    <Xaml:C1TreeViewItem.Header>
      <CheckBox>
        <CheckBox.Content>
          <TextBlock Text="コンピュータ" />
        </CheckBox.Content>
      </CheckBox>
    </Xaml:C1TreeViewItem.Header>
          <Xaml:C1TreeViewItem IsExpanded="True">
      <Xaml:C1TreeViewItem.Header>
        <CheckBox>
          <CheckBox.Content>
          <TextBlock Text="ロカール ディスク (C:)" />
          </CheckBox.Content>
        </CheckBox>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
        </Xaml:C1TreeViewItem.Header>
        <Xaml:C1TreeViewItem>
          <Xaml:C1TreeViewItem.Header>
            <CheckBox>
              <CheckBox.Content>
              <TextBlock Text="Program Files" />
              </CheckBox.Content>
            </CheckBox>
          </Xaml:C1TreeViewItem.Header>
    </Xaml:C1TreeViewItem>
          <Xaml:C1TreeViewItem>
            <Xaml:C1TreeViewItem.Header>
              <CheckBox>
                <CheckBox.Content>
                  <TextBlock Text="ユーザー" />
                </CheckBox.Content>
              </CheckBox>
            </Xaml:C1TreeViewItem.Header>
    </Xaml:C1TreeViewItem>
          <Xaml:C1TreeViewItem>
            <Xaml:C1TreeViewItem.Header>
              <CheckBox>
                <CheckBox.Content>
                  <TextBlock Text="Windows" />
                </CheckBox.Content>
              </CheckBox>
            </Xaml:C1TreeViewItem.Header>
    </Xaml:C1TreeViewItem>
</Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem>
      <Xaml:C1TreeViewItem.Header>
        <CheckBox>
          <CheckBox.Content>
            <TextBlock Text="DVD ドライブ (D:)" />
          </CheckBox.Content>
        </CheckBox>
      </Xaml:C1TreeViewItem.Header>
    </Xaml:C1TreeViewItem>
</Xaml:C1TreeViewItem>
<Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem.Header>
      <CheckBox>
        <CheckBox.Content>
          <TextBlock Text="ネットワーク" />
        </CheckBox.Content>
      </CheckBox>
    </Xaml:C1TreeViewItem.Header>
      </Xaml:C1TreeViewItem>
<Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem.Header>
      <CheckBox>
        <CheckBox.Content>
```


Copyright © GrapeCity, Inc. All rights reserved.

```
            <TextBlock Text="コントロール パネル" />
          </CheckBox.Content>
        </CheckBox>
      </Xaml:C1TreeViewItem.Header>
        </Xaml:C1TreeViewItem>
  <Xaml:C1TreeViewItem>
    <Xaml:C1TreeViewItem.Header>
      <CheckBox>
        <CheckBox.Content>
          <TextBlock Text="ごみ箱" />
        </CheckBox.Content>
      </CheckBox>
    </Xaml:C1TreeViewItem.Header>
    </Xaml:C1TreeViewItem>
  </Xaml:C1TreeViewItem>
</Xaml:C1TreeView>
```

ドラッグ &ドロップの有効化

C1TreeView は、ドラッグ &ドロップ動作をサポートします。詳細については、「ノードのドラッグ &ドロップ」トピックを参照してください。ドラッグ &ドロップ動作を有効にするには、**AllowDragDrop** プロパティを **True** に設定します。

Visual Basic コードの書き方

```
Visual Basic
C1TreeView.AllowDragDrop = True
```

C# コードの書き方

```
C#
C1TreeView.AllowDragDrop = true;
```

ビジュアルなドラッグ &ドロップインジケータを有効にするには、**DragDropArrowMarker** および **DragDropLineMarker** プロパティを設定します。

Copyright © GrapeCity, Inc. All rights reserved.