OrgChart for UWP

2018.03.07 更新

グレープシティ株式会社

OrgChart for UWP

目次

OrgChart for UWP 2
主な特長 3
クイックスタート 4
手順1：C1OrgChart アプリケーションの作成 4
手順2：C1OrgChart コントロールへのコンテンツの追加 4-6
手順3：C1OrgChart アプリケーションの実行 6-7
C1OrgChart の使い方 8
C1OrgChart の要素 8
C1OrgChart のコアプロパティ 8-10
C1OrgChart プロパティでの連結の使用 10-12
高度な連結シナリオ 12-15
レイアウトおよび外観 16
パネル内のレイアウト 16
C1OrgChart のスタイル 16
外観プロパティ 16-17
Orientation 17
FlowDirection 17-18
ChildSpacing 18
Connector 18-19
Alignment 19-20
タスク別ヘルプ 21
アプリケーションへの C1OrgChart の追加 21
C1OrgChart の方向の変更 21-22
C1OrgChart のフロー方向 22
C1OrgChart の項目接続線のカスタマイズ 22
C1OrgChart ノードの展開と折りたたみ 22-32
階層化データテンプレートの使用 33-37

OrgChart for UWP

データの構造や関係を示す階層図を作成できます。**OrgChart for UWP**は、プラットフォームの機能豊富なデータ連結メカニズムを活用して、柔軟で使いやすいコントロールを提供します。

主な特長

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

- **柔軟なデータ連結**
  C1OrgChart コントロールは **ItemsControl** の1つです。このコントロールを、いくつかのサブ項目を保持する1つのエンティティに連結したり、それぞれがいくつかのサブ項目を保持できる項目から成る **IEnumerable** コレクションに連結することができます。
- **多様な方向およびフロー**
  C1OrgChart では、項目を水平または垂直のどちらのフロー方向にも表示できます。コントロールの **Orientation** および FlowDirection プロパティを設定するだけで、組織図のフローを決定できます。
- **折りたたみ可能なノード**
  項目のブランチを非表示にして、コンパクトな表示が可能になります。C1OrgChart のノードには、**TreeView** と同様に、各ノードを展開/折りたたむことができる **IsCollapsed** プロパティがあります。
- **接続線のカスタマイズ**
  C1OrgChart コントロールは、ノードの接続に使用する線をカスタマイズするためのプロパティをいくつか公開しています。これらのプロパティを使用して、接続線の作成に使用するブラシ、太さ、および破線配列をカスタマイズできます。これらのプロパティをデータ項目のプロパティに連結して、関係ごとに線をカスタマイズすることもできます。
- **子の間隔と配置のオプション**
  いくつかのプロパティを設定するだけで、**OrgChart** 内の項目の配置や間隔をカスタマイズできます。このコントロールには、項目間の間隔(ピクセル単位)を制御する **ChildSpacing** プロパティや、見栄えのよさを大きく変える垂直および水平方向の配置プロパティがあります。
- **複雑な階層表示**
  C1OrgChart コントロールは、データテンプレートや連結によって提供される柔軟性に加えて、複雑な階層表示を作成するための高度な連結シナリオもサポートします。条件付き書式設定のように、特定のデータ項目のプロパティに基づいて、一部のノードに異なるテンプレートを使用できます。たとえば、会社の組織図で役員、管理職、事務職を視覚的に区別するために、複数のテンプレートを使用できます。

クイックスタート

このクイックスタートガイドは、**OrgChart 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、**C1OrgChart** コントロールを使用して、簡単なプロジェクトを作成します。新しい Windows ストアアプリケーションを作成し、そのアプリケーションに C1OrgChart コントロールを追加し、C1OrgChart コントロールに表示されるデータを追加します。

手順1：C1OrgChart アプリケーションの作成

この手順では、**OrgChart 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。**C1OrgChart** を使用すると、データの構造や関係を示す階層図を作成できます。プロジェクトをセットアップし、C1OrgChart 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。
3. **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。MainPage.xaml ファイルの XAML ビューを開きます。このクイックスタートでは、XAML マークアップを使用して、いくつかのコントロールを追加します。
4. ツールボックスに移動し、C1OrgChart アイコンをダブルクリックします。これで、アプリケーションにコントロールが追加されます。これで、参照と XAML 名前空間が自動的に追加されます。
5. 次のマークアップを `<OrgChart:C1OrgChart>` タグに追加します。
   - Name="c1OrgChart1"
   - ConnectorThickness="2"
   - ConnectorStroke="Black"
   - Orientation="Vertical"
   - ChildSpacing="20, 30"

   XAML マークアップは次のようになります。

   マークアップ

```
<OrgChart:C1OrgChart Name="c1OrgChart1" ConnectorThickness="2"
ConnectorStroke="Black"
            Orientation="Vertical" ChildSpacing="20,30"
VerticalAlignment="Center" HorizontalAlignment="Center" >
```

6. `<OrgChart:C1OrgChart>` タグと `</OrgChart:C1OrgChart>` タグの間に次のC1OrgChart.ItemTemplate を配置します。

   マークアップ

```
<OrgChart:C1OrgChart.ItemTemplate>
   <DataTemplate>
       <Border Background="#FF6AD400" Width="180" Height="90">
          <TextBlock Text="{Binding Name}" FontSize="20"
HorizontalAlignment="Center"
                VerticalAlignment="Center" Foreground="White"/>
       </Border>
   </DataTemplate>
</OrgChart:C1OrgChart.ItemTemplate>
```

これで、アプリケーションのユーザーインタフェースのセットアップは終了しましたが、C1OrgChart コントロールにはまだコンテンツがありません。次の手順では、C1OrgChart コントロールにコンテンツを追加します。

手順2：C1OrgChart コントロールへのコンテンツの追加

前の手順では、Silverlight アプリケーションを作成し、プロジェクトに C1OrgChart コントロールを追加しました。この手順では、**C1OrgChart** コントロールにコンテンツを追加します。プロジェクトをカスタマイズしてアプリケーションの C1OrgChart コントロールにコンテンツを追加するには、次の手順に従います。

1. ソリューションエクスプローラで、**MainPage.xaml** ファイルを右クリックして[**コードの表示**]を選択します。コードファイルが開きます。
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.OrgChart
```

▶ C# コードの書き方

C#>

```
using C1.Xaml.OrgChart;
```

3. このコードは、アプリケーションにコンテンツを追加します。ページのコンストラクタ内の this.InitializeComponent() メソッド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

```
' 階層を作成します。
Dim uwp As New Platform() With {.Name = "UWP"}
Dim winjs As New Platform() With {.Name = "HTML"}
Dim xaml As New Platform() With {.Name = "XAML"}
Dim dx As New Platform() With {.Name = "DirectX"}
uwp.Subplatforms = New List(Of Platform)()
uwp.Subplatforms.Add(winjs)
uwp.Subplatforms.Add(xaml)
uwp.Subplatforms.Add(dx)
winjs.Subplatforms = New List(Of Platform)()
winjs.Subplatforms.Add(New Platform() With {.Name = "JavaScript"})
xaml.Subplatforms = New List(Of Platform)()
xaml.Subplatforms.Add(New Platform() With {.Name = "C#"})
xaml.Subplatforms.Add(New Platform() With {.Name = "VB"})
dx.Subplatforms = New List(Of Platform)()
dx.Subplatforms.Add(New Platform() With {.Name = "C++"})
' orgchart に設定します。
c1OrgChart1.Header = uwp
```

▶ C# コードの書き方

C#

```
// 階層を作成します
        Platform uwp = new Platform() { Name = "UWP" };
        Platform winjs = new Platform() { Name = "HTML" };
        Platform xaml = new Platform() { Name = "XAML" };
        Platform dx = new Platform() { Name = "DirectX" };

        uwp.Subplatforms = new List<Platform>();
```

```
        uwp.Subplatforms.Add(winjs);
        uwp.Subplatforms.Add(xaml);
        uwp.Subplatforms.Add(dx);

        winjs.Subplatforms = new List<Platform>();
        winjs.Subplatforms.Add(new Platform() { Name = "JavaScript" });

        xaml.Subplatforms = new List<Platform>();
        xaml.Subplatforms.Add(new Platform() { Name = "C#" });
        xaml.Subplatforms.Add(new Platform() { Name = "VB" });

        dx.Subplatforms = new List<Platform>();
        dx.Subplatforms.Add(new Platform() { Name = "C++" });

// orgchart に設定します
        c1OrgChart1.Header = uwp;
    }
  }
```

4. ページのコンストラクタの下に次のクラスを追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
Public Class Platform
    Public Property Name As String
    End Property
    Public Property Subplatforms As IList(Of Platform)
    End Property
End Class
```

▶ C# コードの書き方

C#

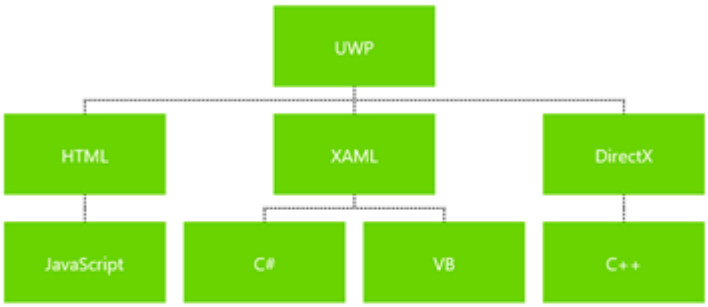
```
public class Platform
    {
        public string Name { get; set; }
        public IList<Platform> Subplatforms { get; set; }
    }
}
```

この手順では、C1OrgChart 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：C1OrgChart アプリケーションの実行

Windows ストアアプリケーションを作成し、**C1OrgChart** コントロールにコンテンツを追加しました。後は、アプリケーションを実行するだけです。

- [**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。アプリケーションは次の図のように表示されます。



おめでとうございます。これで **OrgChart for UWP**のクイックスタートは完了です。簡単な ユニバーサルWindowsアプリケーションを作成し、カスタマイズした **OrgChart for UWP**コントロールを追加しました。

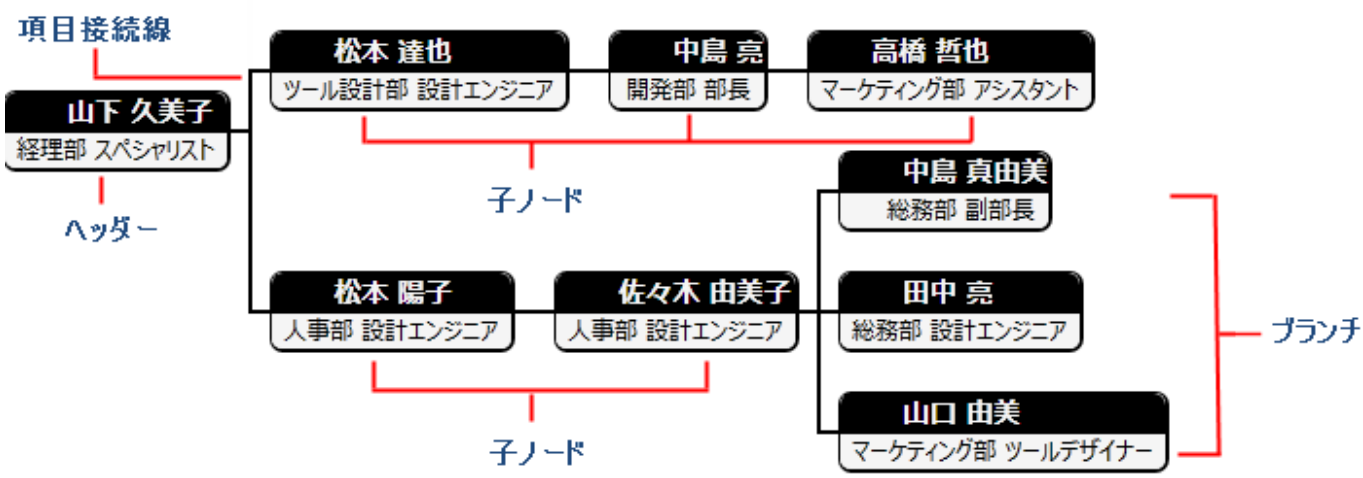
C1OrgChart の使い方

組織図（体制図、系統図）は、組織の構造や、組織内の部署・役職の上下または左右の関係を示す図です。この用語は、ある学問分野や言語グループ内のさまざまな要素を示す図などでも使用されます。

C1OrgChart コントロールを使用すると、あらゆる種類の階層化データを示す組織図を作成できます。このコントロールは、WPF および Silverlight の機能豊富なデータ連結メカニズムを活用して、柔軟で使いやすいツールを提供します。

C1OrgChart の要素

C1OrgChart コントロールは、Header、ChildNodes、**ItemConnector** などの複数の部分で構成されています。次の画像はこれらの各部分を示しています。また、C1OrgChart のブランチの1つも示しています。



C1OrgChart のコアプロパティ

C1OrgChart コントロールは **ItemsControl** の1つです。通常、このコントロールを使用するには、コントロールの **Header** または ItemsSource プロパティを設定し、ItemTemplate プロパティを使用して項目の外観を定義します。

いくつかのサブ項目を含む1つのデータ項目がある場合は、Header プロパティを使用します。いくつかのサブ項目を含む項目のコレクションがある場合は、**ItemsSource** プロパティを使用します。

どちらの方法でも、データ項目にサブ項目を含める必要があります。ほとんどの場合、サブ項目の型はメイン項目と同じになります。たとえば、**Person** クラスがあるとすると、このクラスは、この人に関するいくつかのプロパティと、この親 **Person** の部下にあたる従業員のリストを含む **Subordinates** プロパティを持つことが考えられます。

▶ C# コードの書き方

C#

```
public class Person
    {
      ObservableCollection<Person> _list = new    ObservableCollection<Person>();

        #region ** object model

        public string Name { get; set; }
        public string Position { get; set; }
        public string Notes { get; set; }
        public IList<Person> Subordinates
```

```
        {
            get { return _list; }
        }
```

Header に **Person** オブジェクトを割り当てると、**Subordinates** プロパティに **Person** オブジェクトのコレクションが含まれていることが自動的に検出されます。これだけでデータの階層が確立されます。

データクラスに複数のコレクションプロパティが含まれる場合、またはコレクションが汎用型（**IEnumerable** など）である場合は、ChildItemsPath プロパティを使用して、子（下位）項目を含むプロパティの名前を指定する必要があります。

項目に含まれるサブ項目の型がそれぞれ異なる場合は、HierarchicalDataTemplate を使用して、レベルごとに項目を指定する必要があります。これについては、このドキュメントの後半で説明します。

ItemTemplate コントロールは、データ項目を表示します。これは標準の **DataTemplate** で、XAML で次のように定義できます。

▶ XAML でマークアップの書き方

マークアップ

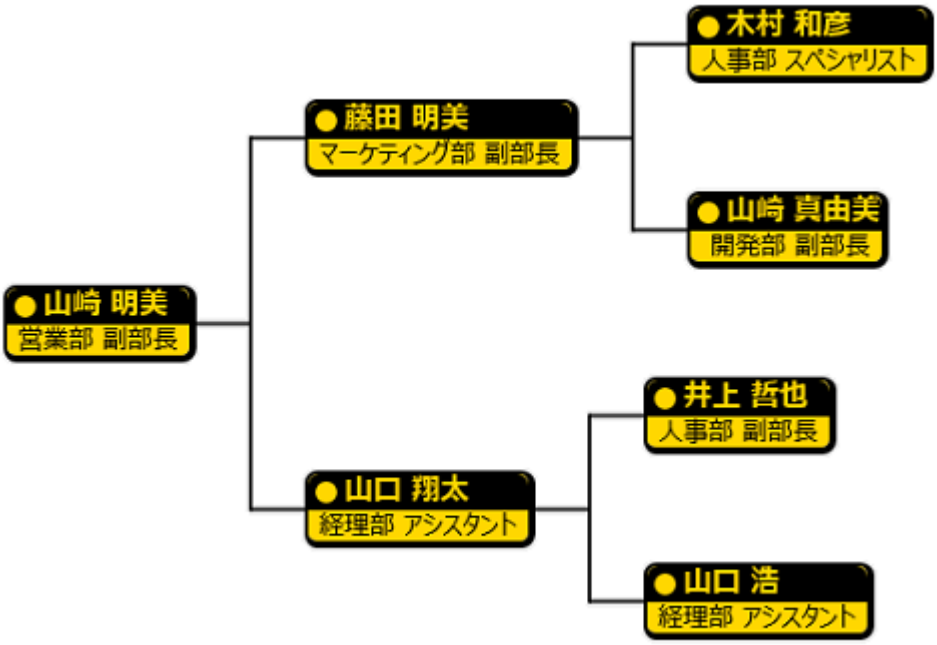
```
<Page.Resources>
        <local:PersonTemplateSelector x:Key="_personTplSelector">
          <local:PersonTemplateSelector.DirectorTemplate>
            <!-- 役員用のデータテンプレート -->
            <DataTemplate>
    <Border Background="Gold" BorderBrush="Black" BorderThickness="2 2 4 4"
CornerRadius="6" Margin="20" MaxWidth="200">
                <StackPanel Orientation="Vertical">
    <Border CornerRadius="6 6 0 0" Background="Black">
                    <StackPanel Orientation="Horizontal">
    <Ellipse Width="12" Height="12" Fill="Gold" Margin="4" />
    <TextBlock Text="{Binding Name}" FontWeight="Bold" FontSize="16"
Foreground="Gold" />
                    </StackPanel>
                  </Border>
    <TextBlock Text="{Binding Position}" Padding="6 0" FontSize="14"
FontStyle="Italic" HorizontalAlignment="Right" />
                </StackPanel>
              </Border>
            </DataTemplate>
          </local:PersonTemplateSelector.DirectorTemplate>
        </local:PersonTemplateSelector>
     </Page.Resources>
```

DataTemplate をリソースとして定義したら、これを **C1OrgChart** コントロールで次のように使用できます。

マークアップ

```
<OrgChart:C1OrgChart Name="_orgChart" Grid.Row="1" ConnectorThickness="2"
ItemTemplateSelector="{StaticResource
_personTplSelector}" Orientation="Horizontal">
```

このテンプレートを多少拡張したバージョンと、ランダムに生成された従業員を使用して作成した組織図の例を下に示します。



C1OrgChart プロパティでの連結の使用

前の例で使用した **ItemTemplate** は、連結を使用して **Employee** クラスのプロパティをビジュアル要素として表示しますが、ビジュアル要素を **C1OrgChart** のプロパティに連結することもできます。

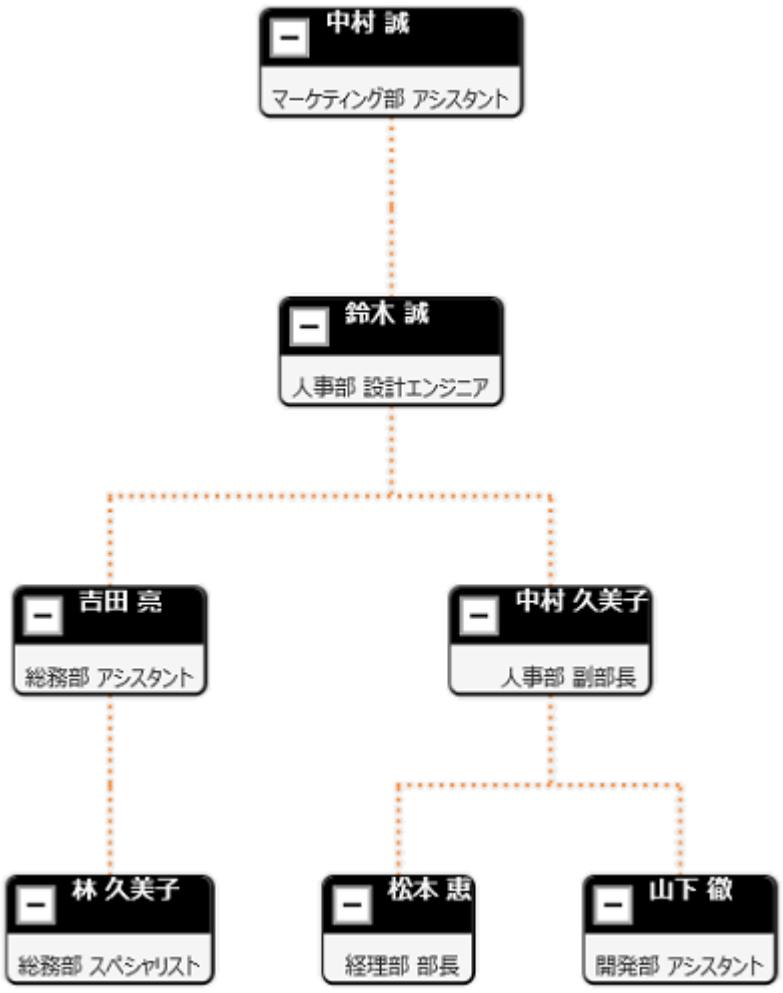
このような例として、**CheckBox.IsChecked** プロパティを **C1OrgChart** の **IsCollapsed** プロパティに連結するとたいへん便利です。これにより、TreeView のように動作する折りたたみ可能な **C1OrgChart** を作成できます。

たとえば、次は、**C1OrgChart** の ItemTemplate プロパティに割り当てたデータテンプレートです。

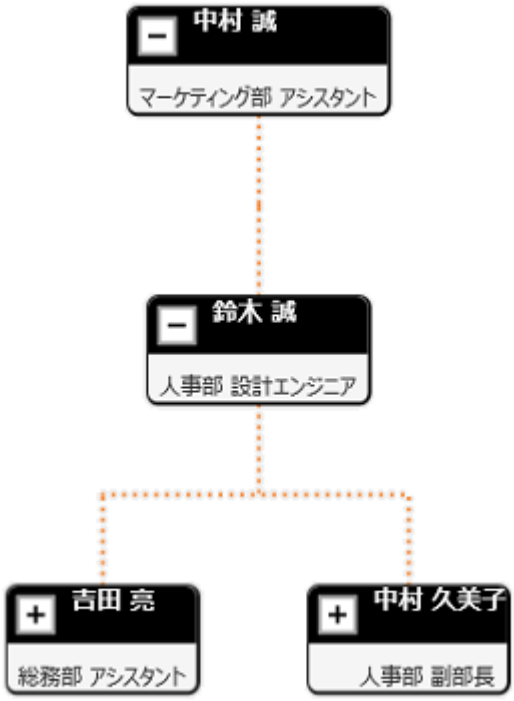
```
<orgchart:C1OrgChart.ItemTemplate>
   <DataTemplate>
   <!-- 外側境界線 -->
     <Border
Background="WhiteSmoke" BorderBrush="Black"
BorderThickness="1 1 2 2" CornerRadius="6"
MaxWidth="200"
<StackPanel Orientation="Vertical">
        <!-- 項目ヘッダー -->
        <Border CornerRadius="6 6 0 0" Background="Black" >
            <StackPanel Orientation="Horizontal">
           <!-- CheckBox を C1OrgChart の IsCollapsed プロパティに連結します -->
            <CheckBox Margin="4 0" Checked="CheckBox_Checked"
Unchecked="CheckBox_Unchecked"/>
               <!-- 項目ヘッダー:人の名前 -->
                <TextBlock Text="{Binding Name}" FontWeight="Bold" FontSize="14"
Foreground="WhiteSmoke" Padding="4 0 0 0" />
            </StackPanel>
          </Border>
          <!-- 本体:人の詳細 -->
```

```
            <TextBlock Text="{Binding Notes}" Padding="6 0" FontSize="9.5"
TextWrapping="Wrap" />
             <TextBlock Text="{Binding Position}" Padding="6 0" FontSize="12"
FontStyle="Italic" HorizontalAlignment="Right" />
          </StackPanel>
        </Border>
      </DataTemplate>
    </orgchart:C1OrgChart.ItemTemplate>
```

この変更により、組織図は次のようになります。



Sarah Williams ノードと Larry Doe ノードのチェックボックスをクリックすると、ブランチが折りたたまれ、コンパクトに表示されます。



高度な連結シナリオ

C1OrgChart コントロールは、データテンプレートや連結によって提供される柔軟性に加えて、2つの標準クラス（**DataTemplateSelector** および **HierarchicalDataTemplate**）を使用する高度な連結シナリオもサポートします。

DataTemplateSelector クラス：このクラスは、データオブジェクトとデータ連結要素に基づいて、異なるテンプレートを選択する方法を提供します。たとえば、役員、管理職、一般社員を、それぞれ異なるテンプレートを使用して表示できます。

それには、**DataTemplateSelector** から継承されるカスタムクラスを作成し、**SelectTemplate** オブジェクトをオーバーライドします。次に、このクラスのインスタンスを作成し、それを C1OrgChart コントロールの **ItemTemplateSelector** プロパティに割り当てます。

次の例は、簡単な **DataTemplateSelector** の実装を示します。

▶ C# コードの書き方

C#

```
/// <summary>
/// 作成される項目のテンプレートを選択するために使用されるクラス。
/// </summary>
 public class PersonTemplateSelector : C1.Xaml.OrgChart.DataTemplateSelector
    {
      public override DataTemplate SelectTemplate(object item, DependencyObject
container)
        {
            var p = item as Person;


            if (p.Position.IndexOf("Director") > -1)
            {
                return DirectorTemplate;
```

```
            }
            else if (p.Position.IndexOf("Manager") > -1)
            {
                return ManagerTemplate;
            }
            else if (p.Position.IndexOf("Designer") > -1)
            {
                return DesignerTemplate;
            }
            else
            {
                return OtherTemplate;
            }
        }
```

カスタム **DataTemplateSelector** を作成したら、通常どおり、XAML で使用できます。

▶ XAML でマークアップの書き方

マークアップ

```
<Page.Resources>
    <!-- テンプレートセレクタ:_tplDirector または _tlpOther を選択します -->
    <local:PersonTemplateSelector x:Key="_personTplSelector">
      <local:PersonTemplateSelector.DirectorTemplate>
        <!-- 役員用のデータテンプレート -->
        <DataTemplate>
          ...
        </DataTemplate>
      </local:PersonTemplateSelector.DirectorTemplate>
      <local:PersonTemplateSelector.ManagerTemplate>
        <!-- 管理職用のデータテンプレート -->
        <DataTemplate>
          ...
        </DataTemplate>
      </local:PersonTemplateSelector.ManagerTemplate>
      <local:PersonTemplateSelector.DesignerTemplate>
        <!-- 設計者用のデータテンプレート -->
        <DataTemplate>
          ...
        </DataTemplate>
      </local:PersonTemplateSelector.DesignerTemplate>
      <local:PersonTemplateSelector.OtherTemplate>
        <!-- その他の従業員用のデータテンプレート -->
        <DataTemplate>
          ...
        </DataTemplate>
      </local:PersonTemplateSelector.OtherTemplate>
    </local:PersonTemplateSelector>
  </Page.Resources>
```

次の図に、この結果を示します。

ItemTemplateSelector は、これらのデータ項目が同じ型である場合に使用できますが、特定のデータ項目のプロパティに基づいて一部の項目を異なる表示にすることもできます。

HierarchicalDataTemplate クラス:このクラスを使用すると、**C1OrgChart** コントロールを複数の型の項目を含む項目に連結できます。たとえば、リーグ、各リーグ内の地区、および各地区内のチームを表示する組織図を作成できます。

それには、サブ項目を含むクラスごとに **HierarchicalDataTemplate** を作成し、階層内の最後のクラス用に通常のデータテンプレートを作成します。この例では、リーグ用と地区用にそれぞれ **HierarchicalDataTemplate** を作成し、さらにチーム用に通常のデータテンプレートを作成します。

▶ XAML でマークアップの書き方

マークアップ

```
<Page.Resources>
        <!-- Team オブジェクト用のテンプレート -->
        <DataTemplate x:Key="TeamTemplate" >
            <Border Background="LightBlue" Padding="4" >
                <TextBlock FontStyle="Italic" Text="{Binding Name}" />
            </Border>
        </DataTemplate>

        <!-- Division オブジェクト用のテンプレート -->
        <Xaml:C1HierarchicalDataTemplate  x:Key="DivisionTemplate"
            ItemTemplate="{StaticResource TeamTemplate}" ItemsSource="{Binding
Teams}">
            <DataTemplate>
                <Border Background="Gold" >
                    <TextBlock Text="{Binding Name}" FontWeight="Bold"
HorizontalAlignment="Center" VerticalAlignment="Center" Padding="20" />
                </Border>
            </DataTemplate>
        </Xaml:C1HierarchicalDataTemplate >
        <!-- League オブジェクト用のテンプレート -->
        <Xaml:C1HierarchicalDataTemplate  x:Key="LeagueTemplate"
            ItemTemplate="{StaticResource DivisionTemplate}" ItemsSource="{Binding
Divisions}">
            <DataTemplate>
                <Border Background="LightCoral" >
                    <TextBlock Text="{Binding Name}" FontWeight="Bold"
HorizontalAlignment="Center" VerticalAlignment="Center" Padding="40" />
                </Border>
            </DataTemplate>
        </Xaml:C1HierarchicalDataTemplate >
```

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

最上位のテンプレートは、**LeagueTemplate** です。**ItemsSource** プロパティは、**League** オブジェクトの表示方法（通常のテンプレート表示）を定義することに加えて、下位オブジェクトを League.Divisions プロパティから取得する必要があることを指定します。最後に、**ItemTemplate** プロパティは、下位オブジェクトの表示に使用されるテンプレートを指定します。

この例では、この **ItemTemplate** は **DivisionTemplate** です。これもまた別の **HierarchicalDataTemplate** です。これは、**Division** オブジェクトの表示方法、下位オブジェクトが **Division.Teams** プロパティによって公開されること、さらにその下位オブジェクトを **TeamTemplate**（通常の非階層化データテンプレート）を使用して表示する必要があることを指定します。

次の点が重要です。

- **HierarchicalDataTemplate** クラスは、標準の **DataTemplate** クラスから派生され、2つのプロパティを追加します。**ItemsSource** は、サブ項目を含むプロパティを指定し、**ItemTemplate** は、下位項目で使用されるテンプレートを指定します。

テンプレートを定義したら、これらを使用するには、通常どおり、ItemTemplate プロパティを設定します。

```
<OrgChart:C1OrgChart Name="_chart" ItemTemplate="{StaticResource LeagueTemplate}"
ConnectorDashArray="1 2" ConnectorStroke="Gray"
HorizontalAlignment="Center" VerticalAlignment="Center" />
```

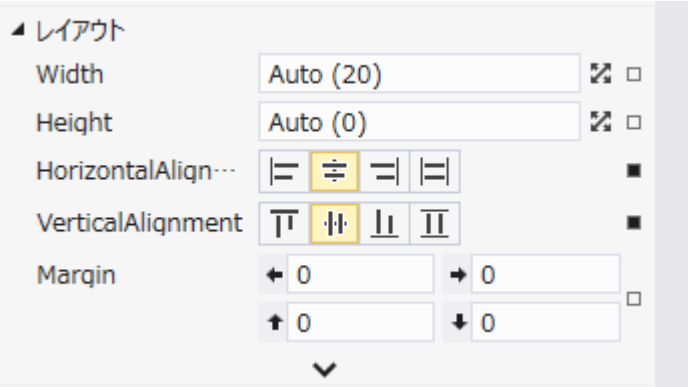
C1OrgChart が階層化データテンプレートを使用して、適切な子コレクションおよびデータテンプレートを選択しながら階層を組み立てていることがわかります。

レイアウトおよび外観

以下のトピックでは、**C1OrgChart** 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。

パネル内のレイアウト

アプリケーションでは、付属するレイアウトプロパティを使用して、**C1OrgChart** や他のコントロールを簡単にレイアウトできます。たとえば、**Grid** パネルでその **Row**、**ColumnSpan**、**RowSpan** の各プロパティを使用してコントロールをレイアウトできます。たとえば、**Grid** パネル内に配置された C1OrgChart コントロールには、次の **Layout** プロパティがあります。



Grid パネル内で、C1OrgChart コントロールのサイズ、配置、および場所を変更できます。

C1OrgChart のスタイル

OrgChart for UWPの C1OrgChart 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
FocusVisualStyle	この要素がキーボードフォーカスを受け取ったときに適用される外観、効果などのスタイル特性をカスタマイズするためのプロパティ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。
GroupStyle	各レベルのグループの外観を定義する **GroupStyle** オブジェクトのコレクションを取得します。
GroupStyleSelector	コレクション内の各グループに **GroupStyle** を適用するためのカスタム選択ロジックを提供するメソッドを取得または設定します。
ItemContainerStyle	各項目に対して生成されたコンテナ要素に適用される **Style** を取得または設定します。
ItemContainerStyleSelector	生成される各コンテナ要素に適用できるスタイルのカスタムスタイル選択ロジックを取得または設定します。
ItemTemplate	各項目の表示に使用される **DataTemplate** を取得または設定します。
ItemTemplateSelector	各項目の表示に使用するテンプレートを選択するためのカスタムロジックを取得または設定します。
Style	この要素のレンダリング時に使用されるスタイルを取得または設定します。これは依存プロパティです。
Template	コントロールテンプレートを取得または設定します。

外観プロパティ

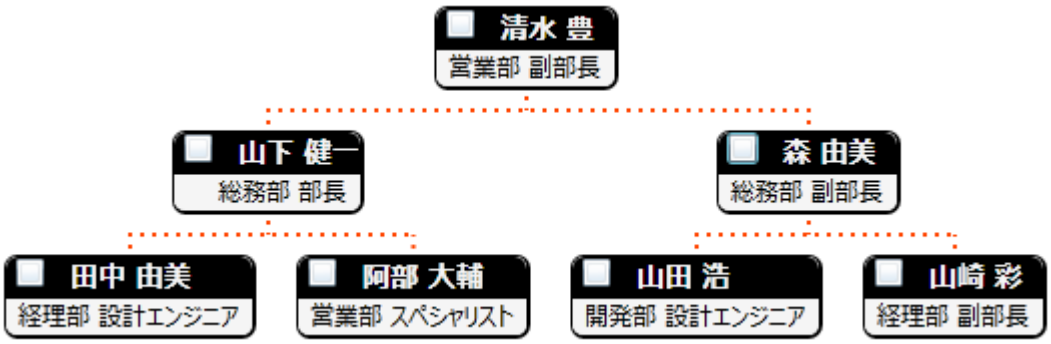
ItemTemplate プロパティを使用して、**C1OrgChart** のノードの外観を完全に自由に指定できます。**C1OrgChart** は、組織図自体の外観をカスタマイズするためのプロパティをいくつか公開します。

Orientation

Orientation プロパティは、組織図のフローを垂直方向または水平方向のどちらにするかを指定できます。デフォルトでは、垂直の **C1OrgChart** が表示されます。**Orientation** プロパティを **Horizontal** に設定すると、次の図のようになります。

マークアップ

```
<OrgChart:C1OrgChart
  Name="_orgChart" Orientation="Horizontal"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
</OrgChart:C1OrgChart>
```

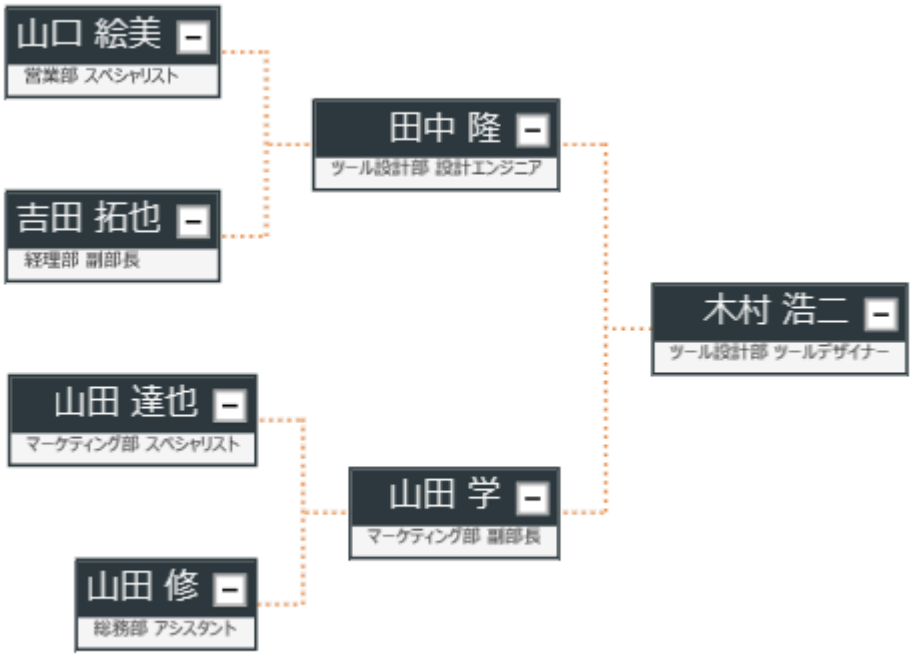


FlowDirection

FlowDirection プロパティでは、組織図のフローを右から左または左から右のどちらにするかを指定できます。デフォルトでは、**C1OrgChart** のフローは左から右になります。**FlowDirection** プロパティを **RightToLeft** に`変更するには、次のマークアップを使用します。

マークアップ

```
<OrgChart:C1OrgChart
  Name="_orgChart" FlowDirection="RightToLeft"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
</OrgChart:C1OrgChart>
```



ChildSpacing

ChildSpacing プロパティでは、項目間の間隔をピクセル単位で制御できます。デフォルト値は "20、20" で、項目の水平方向および垂直方向の間隔がそれぞれ 20 ピクセルになります。たとえば、**ChildSpacing** プロパティを "20、60" に設定すると、次のように表示されます。

マークアップ

```
<OrgChart:C1OrgChart Name="_orgChart" Grid.Row="1" Orientation="Vertical"
ChildSpacing="20, 60" HorizontalAlignment="Center"
VerticalAlignment="Center" ConnectorStroke="OrangeRed" ConnectorThickness="2"
ConnectorDashArray="1 2" Foreground="#FF39B925" />
```

デフォルト設定と比べて、項目間の垂直方向の間隔が広くなっていることがわかります。

また、**ItemTemplate** の **Margin** 値を指定して項目間の間隔を制御することもできます。この方法では、ノード要素の上下左右のスペースをそれぞれ指定できるため、少し柔軟性があります。

Connector

C1OrgChart は、ノードの接続に使用する線をカスタマイズするためのプロパティをいくつか公開しています。**ConnectorStroke**（接続線の作成に使用する **Brush** を指定）、**ConnectorThickness**（線の太さ）、**ConnectorDashArray**（破線の作成に使用）などのプロパティがあります。これらは、**Line** 要素のプロパティに似ています。

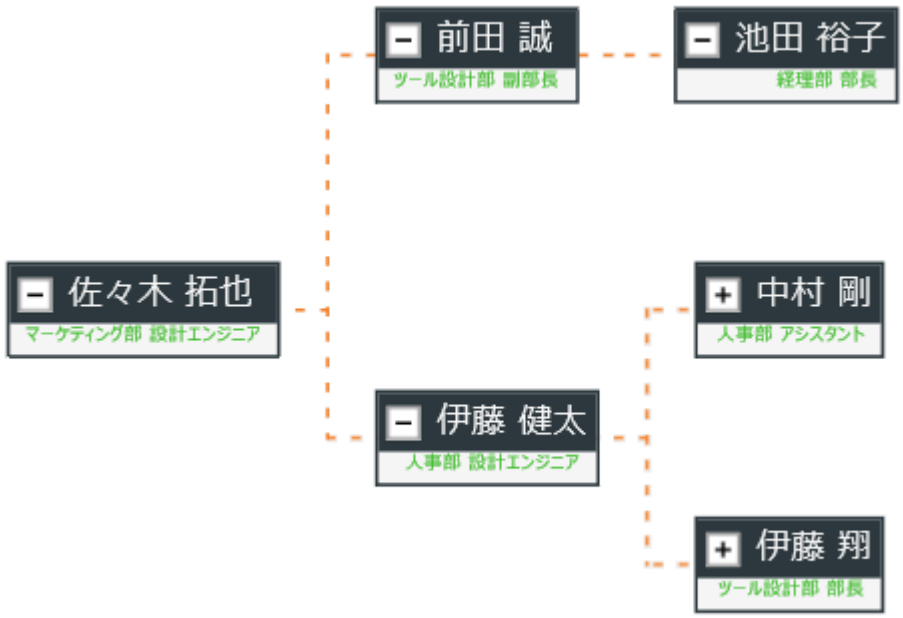
たとえば、灰色の点線を使用して項目を接続する場合は、次の XAML マークアップを使用します。

マークアップ

```
<OrgChart:C1OrgChart Name="_orgChart"
                     ConnectorStroke="OrangeRed"
```

```
            ConnectorThickness="2"
            ConnectorDashArray="3 5"
            Foreground="#FF39B925" />
```

ConnectorDashArray プロパティには、**double** 値のコレクションを使用して、線の部分と空白の部分の長さを ConnectorThickness の単位で指定します。



Alignment

HorizontalContentAlignment および VerticalContentAlignment プロパティを使用して、組織図内のノードの配置をカスタマイズできます。デフォルト値は、どちらのプロパティも **Center** です。この場合、ノードはツリー内の中央に配置されます。その他にも、次のような表示設定が可能です。

Left

HorizontalContentAlignment を "Left" に設定した場合：

マークアップ

```
<OrgChart:C1OrgChart
  Name="_orgChart"
  HorizontalContentAlignment="Left"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
</OrgChart:C1OrgChart>
```

OrgChart は次のように表示されます。

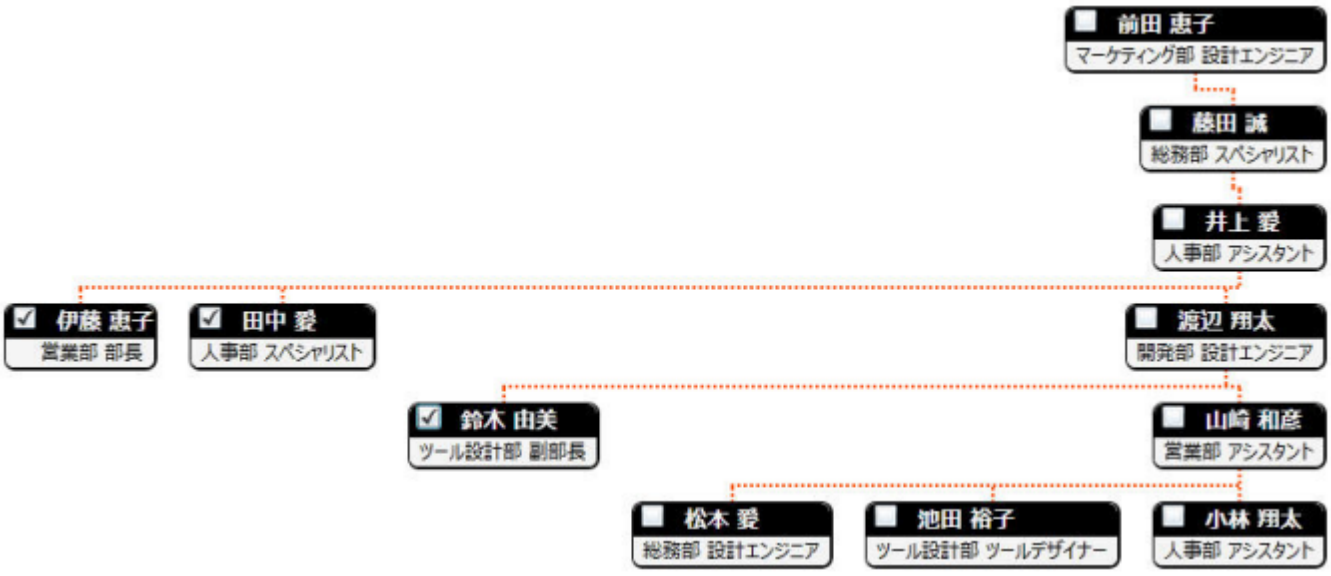
Right

HorizontalContentAlignment を "Right" に設定した場合:

マークアップ

```
<OrgChart:C1OrgChart
  Name="_orgChart"
  HorizontalContentAlignment="Right"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
</OrgChart:C1OrgChart>
```

OrgChart は次のように表示されます。



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 Visual Studio に精通しており、C1OrgChart 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。**OrgChart 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、**OrgChart 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、ほとんどのタスク別ヘルプトピックは、Windows ストアアプリケーションが作成されており、そのプロジェクトに C1OrgChart 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については、「**アプリケーションへの C1OrgChart の追加**」を参照してください。

アプリケーションへの C1OrgChart の追加

C1OrgChart 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 2012 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. ツールボックスに移動し、C1OrgChart アイコンをダブルクリックします。これで、アプリケーションにコントロールが追加されます。これで、参照と XAML 名前空間が自動的に追加されます。
4. ページの Grid タグ内に`<OrgChart:C1OrgChart Name="C1OrgChart1" />` タグを追加して、アプリケーションに C1OrgChart コントロールを追加します。
   XAML は次のようになります。

   マークアップ

   ```
   <Grid Name="LayoutRoot" Background="White">
       <OrgChart:C1OrgChart Name="C1OrgChart1" />
   </Grid>
   ```

   これで、"C1OrgChart1" という名前の C1OrgChart コントロールがアプリケーションに追加されます。ここでアプリケーションを実行すると、空白のページが表示されます。

これで、アプリケーションのユーザーインタフェースが正しくセットアップされましたが、このアプリケーションを実行すると、C1OrgChart コントロールにコンテンツがないことがわかります。

> **メモ: C1OrgChart** コントロールが Visual Studio のツールボックスにインストールされている場合は、アイコンをダブルクリックするか、ページにコントロールをドラッグするだけで、上のコントロール関連手順が自動的に実行されます。

C1OrgChart の方向の変更

C1OrgChart は、水平方向または垂直方向のフローで表示できます。このプロパティは、XAML または[プロパティ]ウィンドウで設定できます。

XAML の場合

`Orientation=Vertical` を `<OrgChart:C1OrgChart>` 開始タグに挿入します。C1OrgChart コントロールの XAML マークアップは次のようになります。

マークアップ

```
<OrgChart:C1OrgChart Name="_orgChart" Orientation="Horizontal">
```

[プロパティ]ウィンドウの場合

次の手順に従って、C1OrgChart の Orientation プロパティを変更します。

1. [プロパティ]ウィンドウで Orientation プロパティを見つけます。
2. ドロップダウンリストを使用して、この値を "**Vertical**" に変更します。

C1OrgChart のフロー方向

FlowDirection プロパティを使用して、組織図を右から左または左から右のどちらの方向に表示するかを指定できます。

XAML の場合

`<OrgChart:C1OrgChart>` 開始タグを見つけて、タグに `FlowDirection="RightToLeft"` を挿入します。`<OrgChart:C1OrgChart>` マークアップは次のようになります。

マークアップ

```
<OrgChart:C1OrgChart Name="_orgChart" Orientation="Horizontal"
FlowDirection="RightToLeft">
```

[プロパティ]ウィンドウの場合

1. [プロパティ]ウィンドウで **FlowDirection** プロパティを見つけます。
2. ドロップダウンリストを使用して、この値を "**RightToLeft**" に変更します。

C1OrgChart の項目接続線のカスタマイズ

ConnectorStroke、**ConnectorThickness**、**ConnectorDashArray** などのプロパティを使用して、C1OrgChart のノードの接続に使用する線をカスタマイズできます。これらのプロパティは、XAML マークアップまたはデザインビューの[プロパティ]ウィンドウで設定できます。

XAML の場合

項目接続線の色を変更するには、`ConnectorStroke="#FF970014"` を `<OrgChart:C1OrgChart>` 開始タグに挿入します。

項目接続線の太さを変更するには、ConnectorStroke マークアップの後に `ConnectorThickness="3"` を挿入します。

使用される項目接続線のタイプをカスタマイズするには、ConnectorThickness マークアップの後に `ConnectorDashArray="1 1"` を挿入します。これにより、破線の接続線が作成されます。

最終的な XAML マークアップは次のようになります。

マークアップ

```
<OrgChart:C1OrgChart Name="_orgChart" Orientation="Horizontal"
ConnectorStroke="#FF970014" ConnectorThickness="2" ConnectorDashArray="1 1">
```

[プロパティ]ウィンドウの場合

デザインビューの[プロパティ]ウィンドウで、項目接続線をカスタマイズすることもできます。

1. ConnectorStroke プロパティを見つけ、カラーピッカーを使用して項目接続線の新しい色を選択します。
2. ConnectorThickness プロパティを見つけ、新しい太さを選択します。このヘルプでは、"3" を使用します。
3. [F5]キーを押してアプリケーションを実行し、接続線が変更されていることを確認します。C1OrgChart コントロールは次の図のように表示されます。

C1OrgChart ノードの展開と折りたたみ

C1OrgChart では、**TreeView** コントロールと同様に動作する折りたたみ可能な C1OrgChart を作成できます。C1OrgChart ノードを展開/折りたたむには、次の手順に従います。

1. 次のマークアップを挿入して、C1OrgChart コントロールとそのコントロールパネルを作成します。次の XAML は、C1OrgChart コントロールに加えて、**ScrollViewer** コントロールを追加します。

   ▶ XAML でマークアップの書き方

マークアップ

```
<!-- 組織図 -->
<ScrollViewer Grid.Row="2" HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto">
<orgchart:C1OrgChart
x:Name="_orgChart"
Grid.Row="1"
Orientation="Vertical"
HorizontalAlignment="Center" VerticalAlignment="Center"
ConnectorStroke="OrangeRed" ConnectorThickness="{Binding Path=Subordinates.Count}"
Foreground="#FF39B925" Xaml:C1NagScreen.Nag="True" >

         <!-- スケール変換をスライダに連結する -->
        <orgchart:C1OrgChart.RenderTransform>
        <ScaleTransform
                ScaleX="{Binding Value, ElementName=_sliderZoom}"
                ScaleY="{Binding Value, ElementName=_sliderZoom}" />
        </orgchart:C1OrgChart.RenderTransform>

        <!-- ツリーノードの表示に使用されるテンプレート -->
        <orgchart:C1OrgChart.ItemTemplate>
        <DataTemplate>

        <!-- 外側境界線 -->
        <Border
                      Background="WhiteSmoke" BorderBrush="Black"
                      BorderThickness="1 1 2 2" CornerRadius="6"
                      MaxWidth="200" >
            <StackPanel Orientation="Vertical" >

                <!-- 項目ヘッダー -->
                <Border CornerRadius="6 6 0 0" Background="Black" >
                    <StackPanel Orientation="Horizontal">

        <!-- CheckBox を C1OrgChart の IsCollapsed プロパティに連結します -->
        <CheckBox Margin="4 0" Checked="CheckBox_Checked"
Unchecked="CheckBox_Unchecked"/>

        <!-- 項目ヘッダー:人の名前 -->
        <TextBlock Text="{Binding Name}" FontWeight="Bold" FontSize="14"
Foreground="WhiteSmoke"
                                Padding="4 0 0 0" />
            </StackPanel>
        </Border>
```

```
        <!-- 本体:人の詳細 -->
        <TextBlock Text="{Binding Notes}" Padding="6 0" FontSize="9.5"
TextWrapping="Wrap" />
        <TextBlock Text="{Binding Position}" Padding="6 0" FontSize="12"
FontStyle="Italic"
                HorizontalAlignment="Right" />
    </StackPanel>
</Border>
</DataTemplate>
</orgchart:C1OrgChart.ItemTemplate>
</orgchart:C1OrgChart>
</ScrollViewer>
```

2. 次の XAML マークアップを <Grid> タグの下で <Scrollviewer> タグの前に追加します。これはいくつかの Grid 行定義、アプリケーションのタイトル、ボタンとスライダを含むコントロールパネルを追加します。このコントロールパネルを使用して、**C1OrgChart** のズーム、方向、データを制御できます。

▶ XAML でマークアップの書き方

マークアップ

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

        <!-- サンプルタイトル -->
        <StackPanel Orientation="Horizontal" >
            <TextBlock Text="C1OrgChart: Collapse/Expand Sample" FontSize="16"
                VerticalAlignment="Bottom" Foreground="Black" />
            <TextBlock x:Name="_tbTotal" VerticalAlignment="Bottom" />
        </StackPanel>

        <!-- コントロールパネル -->
        <StackPanel Orientation="Horizontal" VerticalAlignment="Top" Grid.Row="1">
            <Button Content="データを更新" Padding="8 0" Click="Button_Refresh_Click"
                Foreground="Black" BorderBrush="Black" />
            <TextBlock Text=" ズーム: " VerticalAlignment="Center" />
            <Slider x:Name="_sliderZoom" Minimum=".01" Maximum="1" Value="1"
Width="100"
                SmallChange="0.01" StepFrequency="0.01" Height="44"
HorizontalAlignment="Left"
                Foreground="#FFFF1B1B" Background="#29C91515"
VerticalAlignment="Center" />
            <CheckBox Margin="20 0" Content="水平方向" Click="CheckBox_Click"
                Background="#FF0Cffff" Foreground="Black" />
            <Button Content="3 レベルまで展開する" Padding="8 0"
                Click="Button_Collapse_Click" Foreground="Black" BorderBrush="Black"
/>
        </StackPanel>
```

3. 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。次の名前空間をコードファイルにインポートします。

▶ Visual Basic コードの書き方

Visual Basic

```
Imports C1.Xaml.OrgChart
```

▶ C# コードの書き方

C#
```
using C1.Xaml.OrgChart
```

4. **InitializeComponent()** メソッドのすぐ下に、次のコードを挿入します。

▶ Visual Basic コードの書き方

Visual Basic
```
CreateData()
```

▶ C# コードの書き方

C#
```
CreateData();
```

5. 次のように、ボタンクリックイベントとチェックボックスクリックイベントを処理し、**C1OrgChart** の切り替えを処理するコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic
```
Private Sub CheckBox_Click(sender As Object, e As RoutedEventArgs)
        _orgChart.Orientation = If(DirectCast(sender, CheckBox).IsChecked.Value,
                        Orientation.Horizontal, Orientation.Vertical)
    End Sub
    ' 新しいランダムデータを使用して組織図を再構築します
    Private Sub Button_Refresh_Click(sender As Object, e As RoutedEventArgs)
        CreateData()
    End Sub
    ' 組織図をレベル3まで折りたたみます
    Private Sub Button_Collapse_Click(sender As Object, e As RoutedEventArgs)
        ToggleCollapseExpand(_orgChart, 0, 3)
    End Sub
    ' 組織図を指定されたレベルまで折りたたみます
    Private Sub ToggleCollapseExpand(node As C1OrgChart, level As Integer,
        maxLevel As Integer)
        If level >= maxLevel Then
            node.IsCollapsed = True
        Else
            node.IsCollapsed = False
            For Each subNode In node.ChildNodes
                ToggleCollapseExpand(subNode, level + 1, maxLevel)
            Next
        End If
    End Sub
```

▶ C# コードの書き方

C#
```
void CheckBox_Click(object sender, RoutedEventArgs e)
        {
            _orgChart.Orientation = ((CheckBox)sender).IsChecked.Value
                ? Orientation.Horizontal
```

```
                : Orientation.Vertical;
        }
        // 新しいランダムデータを使用して組織図を再構築します
        void Button_Refresh_Click(object sender, RoutedEventArgs e)
        {
            CreateData();
        }
        // 組織図をレベル3まで折りたたみます
        void Button_Collapse_Click(object sender, RoutedEventArgs e)
        {
            ToggleCollapseExpand(_orgChart, 0, 3);
        }
        // 組織図を指定されたレベルまで折りたたみます
        void ToggleCollapseExpand(C1OrgChart node, int level, int maxLevel)
        {
            if (level >= maxLevel)
            {
                node.IsCollapsed = true;
            }
            else
            {
                node.IsCollapsed = false;
                foreach (var subNode in node.ChildNodes)
                {
                    ToggleCollapseExpand(subNode, level + 1, maxLevel);
                }
            }
        }
```

6. 次のように、新しいランダムデータを作成し、C1OrgChart を展開/折りたたむためのチェックボックスイベントを処理するコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
    ' ランダムデータを作成し、それを組織図に割り当てます
    Private Sub CreateData()
        Dim p = Person.CreatePerson(15)
        _tbTotal.Text = String.Format(" ({0} items total)", p.TotalCount)
        _orgChart.Header = p
    End Sub
    ' OrgChart サンプルに展開/折りたたみのサポートを追加します
    Private Sub CheckBox_Unchecked(sender As Object, e As RoutedEventArgs)
        CheckedChanged(sender)
    End Sub
    Private Sub CheckBox_Checked(sender As Object, e As RoutedEventArgs)
        CheckedChanged(sender)
    End Sub
    Private Sub CheckedChanged(sender As Object)
Dim checkBox As CheckBox = TryCast(sender, CheckBox)
Dim parent As FrameworkElement = TryCast(VisualTreeHelper.GetParent(checkBox),
        FrameworkElement)
While parent IsNot Nothing AndAlso Not (TypeOf parent Is C1OrgChart)
    parent = TryCast(VisualTreeHelper.GetParent(parent), FrameworkElement)
End While
If parent IsNot Nothing Then
```

```
    Dim orgChart As C1OrgChart = TryCast(parent, C1OrgChart)
    If checkBox.IsChecked IsNot Nothing Then
        orgChart.IsCollapsed = checkBox.IsChecked.Value
            End If
        End If
    End Sub
```

▶ C# コードの書き方

C#

```
// ランダムデータを作成し、それを組織図に割り当てます
        void CreateData()
        {
            var p = Person.CreatePerson(15);
            _tbTotal.Text = string.Format(" ({0} items total)", p.TotalCount);
            _orgChart.Header = p;
        }
        // OrgChart サンプルに展開/折りたたみのサポートを追加します
        private void CheckBox_Unchecked(object sender, RoutedEventArgs e)
        {
           CheckedChanged(sender);
        }
private void CheckBox_Checked(object sender, RoutedEventArgs e)
        {
            CheckedChanged(sender);
        }
        private void CheckedChanged(object sender)
        {
            CheckBox checkBox = sender as CheckBox;
            FrameworkElement parent = VisualTreeHelper.GetParent(checkBox)
                    as FrameworkElement;
            while (parent != null && !(parent is C1OrgChart))
            {
                parent = VisualTreeHelper.GetParent(parent) as FrameworkElement;
            }
            if (parent != null)
            {
                C1OrgChart orgChart = parent as C1OrgChart;
                if (checkBox.IsChecked != null)
                {
                    orgChart.IsCollapsed = checkBox.IsChecked.Value;
                }
            }
        }
    }
}
```

7. ソリューションエクスプローラで、アプリケーション名を見つけます。名前を右クリックし、リストから[**追加**]→[**新しい項目**]を選択します。テンプレートウィンドウで[**コードファイル**]を選択し、コードファイル名を "**Person.cs**" または "**Person.vb**" と指定します。
8. 次の名前空間を **Person** コードファイルに追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
Imports System
```

```
Imports System.Collections
Imports System.Collections.Generic
Imports System.Collections.ObjectModel
```

▶ C# コードの書き方

C#

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
```

9. 名前空間の下に次のコードを挿入して、C1OrgChart のデータを作成するために呼び出される階層化データ項目を作成します。

▶ Visual Basic コードの書き方

Visual Basic

```
Namespace Data
    ''' <summary>
    ''' 階層化データ項目:Person は型 Person の Subordinates を持つ。
    ''' </summary>
    PublicClassPerson
        Private _list AsNewObservableCollection(OfPerson)()
#Region"** object model"
        PublicProperty Name() AsString
            Get
                Return m_Name
            EndGet
            Set(value AsString)
                m_Name = Value
            EndSet
        EndProperty
        Private m_Name AsString
        PublicProperty Position() AsString
            Get
                Return m_Position
            EndGet
            Set(value AsString)
                m_Position = Value
            EndSet
        EndProperty
        Private m_Position AsString
        PublicProperty Notes() AsString
            Get
                Return m_Notes
            EndGet
            Set(value AsString)
                m_Notes = Value
            EndSet
        EndProperty
        Private m_Notes AsString
        PublicReadOnlyProperty Subordinates() AsIList(OfPerson)
            Get
                Return _list
            EndGet
```

```
        EndProperty
        PublicReadOnlyProperty TotalCount() AsInteger
            Get
                Dim count = 1
                ForEach p In Subordinates
                    count += p.TotalCount
                Next
                Return count
            EndGet
        EndProperty
        PublicOverridesFunction ToString() AsString
            ReturnString.Format("{0}:" & vbCr & vbLf & vbTab & "{1}", Name, Position)
        EndFunction
#EndRegion
#Region"** Person factory"
Shared _rnd AsNewRandom()
Shared _positions AsString() = "アシスタント|部長|ツールデザイナー|設計エンジニア|
        副部長|スペシャリスト".Split("|"c)
Shared _areas AsString() = "開発部|ツール設計部|営業部|マーケティング部|人事部|
        経理部|総務部".Split("|"c)
Shared _first AsString() = "浩|誠|健一|大輔|達也|翔太|浩一|哲也|剛|大介|
        健太|拓也|豊|修|和彦|学|直樹|健太郎|浩二|徹|隆|亮|翔|恵子|久美子|由美子|
        明美|直美|陽子|智子|絵美|恵|裕子|愛|真由美|由美|麻衣|美穂|愛美|彩".Split("|"c)
Shared _last AsString() = "佐藤|鈴木|高橋|田中|伊藤|山本|渡辺|中村|小林|
        加藤|吉田|山田|佐々木|山口|松本|井上|木村|斎藤|林|清水|山崎|阿部|森|池田|
        橋本|山下|石川|中島|前田|藤田".Split("|"c)
Shared _verb AsString() = "likes|reads|studies|hates|exercises|dreams|
        plays|writes|argues|sleeps|ignores".Split("|"c)
Shared _adjective AsString() = "long|short|important|pompous|hard|complex|
        advanced|modern|boring|strange|curious|obsolete|bizarre".Split("|"c)
Shared _noun AsString() = "products|tasks|goals|campaigns|books|computers|
        people|meetings|food|jokes|accomplishments|screens|pages".Split("|"c)
PublicSharedFunction CreatePerson(level AsInteger) AsPerson
    Dim p = CreatePerson()
    If level > 0 Then
        level -= 1
        For i AsInteger = 0 To _rnd.[Next](1, 4) - 1
            p.Subordinates.Add(CreatePerson(_rnd.[Next](level \ 2, level)))
        Next
    EndIf
    Return p
EndFunction
PublicSharedFunction CreatePerson() AsPerson
    Dim p = NewPerson()
    p.Position = String.Format("{0} of {1}", GetItem(_positions),
        GetItem(_areas))
    p.Name = String.Format("{0} {1}", GetItem(_first), GetItem(_last))
    p.Notes = String.Format("{0} {1} {2} {3}", p.Name, GetItem(_verb),
        GetItem(_adjective), GetItem(_noun))
    While _rnd.NextDouble() < 0.5
        p.Notes += String.Format(" and {0} {1} {2}", GetItem(_verb),
        GetItem(_adjective), GetItem(_noun))
    EndWhile
    p.Notes += "."
    Return p
EndFunction
```

```
PrivateSharedFunction GetItem(list AsString()) AsString
    Return list(_rnd.[Next](0, list.Length))
EndFunction
#EndRegion
    EndClass
EndNamespace
```

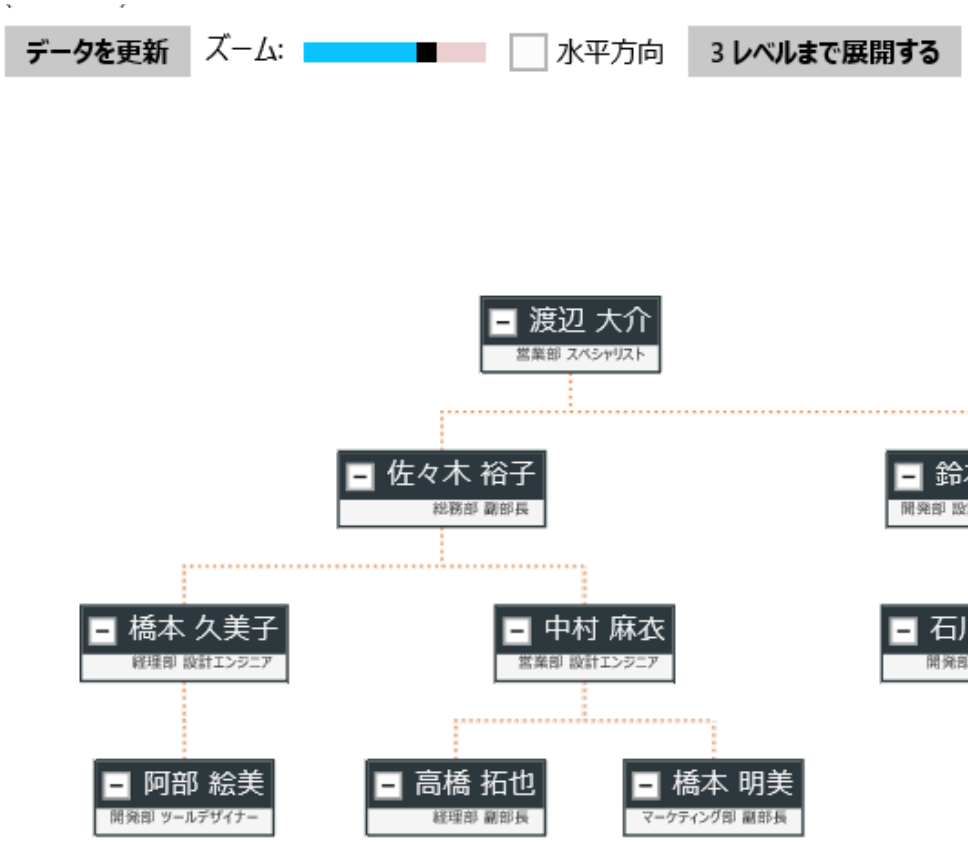
▶ C# コードの書き方

C#

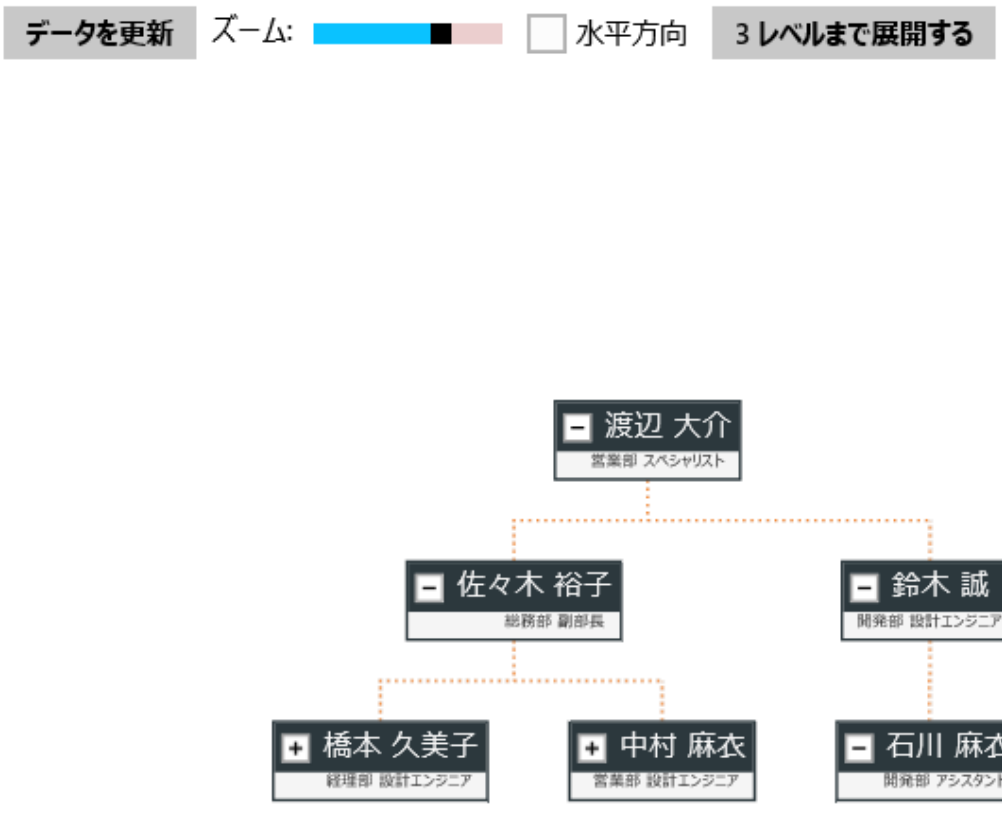
```
namespace CollapseExpand
{
    /// <summary>
    /// 階層化データ項目:Person は型 Person の Subordinates を持つ。
    /// </summary>
    public class Person
    {
ObservableCollection<Person> _list = new ObservableCollection<Person>();
#region ** object model
public string Name { get; set; }
public string Position { get; set; }
public string Notes { get; set; }
public IList<Person> Subordinates
{
    get { return _list; }
}
public int TotalCount
{
    get
    {
        var count = 1;
        foreach (var p in Subordinates)
        {
            count += p.TotalCount;
        }
        return count;
    }
}
public override string ToString()
{
    return string.Format("{0}:\r\n\t{1}", Name, Position);
}
#endregion
#region ** Person factory
static Random _rnd = new Random();
static string[] _positions = "アシスタント|部長|ツールデザイナー|設計
        エンジニア|副部長|スペシャリスト".Split('
static string[] _areas = "開発部|ツール設計部|営業部|マーケティング部|
        人事部|経理部|総務部".Split('
static string[] _first = "浩|誠|健一|大輔|達也|翔太|浩一|哲也|剛|
        大介|健太|拓也|豊|修|和彦|学|直樹|健太郎|浩二|徹|隆|亮|翔|恵子|
        久美子|由美子|明美|直美|陽子|智子|絵美|恵|裕子|愛|真由美|由美|麻衣|
        美穂|愛美|彩".Split('
static string[] _last = "佐藤|鈴木|高橋|田中|伊藤|山本|渡辺|中村|
        小林|加藤|吉田|山田|佐々木|山口|松本|井上|木村|斎藤|林|清水|山崎|
```

```
        阿部|森|池田|橋本|山下|石川|中島|前田|藤田".Split('
static string[] _verb = "likes|reads|studies|hates|exercises|
        dreams|plays|writes|argues|sleeps|ignores".Split('
static string[] _adjective = "long|short|important|pompous|hard|
        complex|advanced|modern|boring|strange|curious|obsolete|
        bizarre".Split('
static string[] _noun = "products|tasks|goals|campaigns|books|
        computers|people|meetings|food|jokes|accomplishments|screens|
        pages".Split('
public static Person CreatePerson(int level)
{
    var p = CreatePerson();
    if (level > 0)
    {
        level--;
        for (int i = 0; i < _rnd.Next(1, 4); i++)
        {
            p.Subordinates.Add(CreatePerson(_rnd.Next(level / 2, level)));
        }
    }
    return p;
}
public static Person CreatePerson()
{
    var p = new Person();
    p.Position = string.Format("{0} of {1}", GetItem(_positions),
        GetItem(_areas));
    p.Name = string.Format("{0} {1}", GetItem(_first), GetItem(_last));
    p.Notes = string.Format("{0} {1} {2} {3}", p.Name, GetItem(_verb),
        GetItem(_adjective), GetItem(_noun));
    while (_rnd.NextDouble() < .5)
    {
        p.Notes += string.Format(" and {0} {1} {2}", GetItem(_verb),
        GetItem(_adjective), GetItem(_noun));
    }
    p.Notes += ".";
    return p;
}
static string GetItem(string[] list)
{
    return list[_rnd.Next(0, list.Length)];
}
        #endregion
    }
}
```

10. [F5]キーを押してアプリケーションを実行します。C1OrgChart は次の図のようになります。



メイン項目ノードの隅にあるチェックボックスをクリックします。C1OrgChart が折りたたまれることを確認します。



階層化データテンプレートの使用

このトピックでは、**DataTemplateSelector** および **HierarchicalDataTemplate** クラスを使用した高度な連結シナリオについて説明します。このトピックは、Windows ストアアプリケーションが作成されており、アプリケーションに適切な参照が追加されていることを前提とします。

1. 次の XAML マークアップを名前空間宣言の下に追加して、データテンプレートを作成します。

▶ XAML でマークアップの書き方

マークアップ

```
<Page.Resources>
<!-- Team オブジェクト用のテンプレート -->
<DataTemplate x:Key="TeamTemplate" >
<Border Background="LightBlue" Padding="4" >
    <TextBlock FontStyle="Italic" Text="{Binding Name}" />
</Border>
</DataTemplate>

<!-- Division オブジェクト用のテンプレート -->
<Xaml:C1HierarchicalDataTemplate  x:Key="DivisionTemplate"
ItemTemplate="{StaticResource TeamTemplate}" ItemsSource="{Binding Teams}">
<DataTemplate>
    <Border Background="Gold" >
        <TextBlock Text="{Binding Name}" FontWeight="Bold"
HorizontalAlignment="Center" VerticalAlignment="Center" Padding="20" />
    </Border>
</DataTemplate>
</Xaml:C1HierarchicalDataTemplate >
<!-- League オブジェクト用のテンプレート -->
<Xaml:C1HierarchicalDataTemplate  x:Key="LeagueTemplate"
ItemTemplate="{StaticResource DivisionTemplate}"
ItemsSource="{Binding Divisions}">
<DataTemplate>
<Border Background="LightCoral" >
    <TextBlock Text="{Binding Name}" FontWeight="Bold"
HorizontalAlignment="Center" VerticalAlignment="Center" Padding="40" />
</Border>
</DataTemplate>
</Xaml:C1HierarchicalDataTemplate >
</Page.Resources>
```

2. 次の XAML マークアップを挿入して、グリッドレイアウト、C1OrgChart コントロール、および **ScrollViewer** コントロールを作成します。

▶ XAML でマークアップの書き方

マークアップ

```
<Grid x:Name="LayoutRoot" Background="White">
<Grid.RowDefinitions>
<RowDefinition Height="Auto" />
<RowDefinition Height="118*" />
<RowDefinition Height="158*" />
</Grid.RowDefinitions>
```

```
 <!-- サンプルタイトル -->
 <StackPanel Orientation="Horizontal" >
 <TextBlock Text="C1OrgChart: HierarchicalDataTemplate" FontSize="16"
VerticalAlignment="Bottom" />
 <TextBlock x:Name="_tbTotal" VerticalAlignment="Bottom" />
 </StackPanel>

 <ScrollViewer Grid.Row="1" HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto" Padding="0" >
 <OrgChart:C1OrgChart
 Name="_chart" ItemTemplate="{StaticResource
LeagueTemplate}"
 ConnectorDashArray="1 2"
ConnectorStroke="Gray"
 HorizontalAlignment="Center"
VerticalAlignment="Center" />
 </ScrollViewer>
 <Xaml:C1TreeView Name="_tree" Grid.Row="2"
 ItemTemplate="{StaticResource LeagueTemplate}"/>
 </Grid>
```

3. ツールボックスで C1TreeView コントロールを見つけ、アプリケーションの `<ScrollViewer> </ScrollViewer>` タグの下に追加します。`<Xaml:C1TreeView>` タグに次のコードを挿入します。

```

Name="_tree" Grid.Row="2" ItemTemplate="{StaticResource LeagueTemplate}"
```

4. アプリケーションを右クリックしてコードビューに切り替え、リストから[**コードの表示**]を選択します。
5. 次の名前空間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.OrgChart
```

▶ C# コードの書き方

C#

```
using C1.Xaml.OrgChart;
```

6. **InitializeComponent()** メソッドのすぐ下に、次のコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
' データオブジェクトを作成します
Dim league__1 = League.GetLeague()
' それを C1OrgChart で表示します
_chart.Header = league__1
' 次のコードでも同じです
'_chart.ItemsSource = new object[] { league };
' それを TreeView で表示します
_tree.ItemsSource = New Object() {league__1}
```

▶ C# コードの書き方

C#

```
// データオブジェクトを作成します
 var league = League.GetLeague();
 // それを C1OrgChart で表示します
 _chart.Header = league;
 // 次のコードでも同じです
 // _chart.ItemsSource = new object[] { league };
 // それを TreeView で表示します
 _tree.ItemsSource = new object[] { league };
```

7. 次のコードを挿入して、C1OrgChart と **C1TreeView** コントロールに表示されるチーム、リーグ、地区を作成します。

▶ C# コードの書き方

C#

```
public class League
    {
        public string Name { get; set; }
        public List<Division> Divisions { get; set; }
        public static League GetLeague()
        {
            var league = new League();
            league.Name = "主要リーグ";
            league.Divisions = new List<Division>();
            foreach (var div in "北部,南部,東部,西部".Split(','))
            {
                var d = new Division();
                league.Divisions.Add(d);
                d.Name = div;
                d.Teams = new List<Team>();
                foreach (var team in "第1地区,第2地区,第3地区,第4地区".Split(','))
                {
                    var t = new Team();
                    d.Teams.Add(t);
                    t.Name = string.Format("{0} {1}", team, div);
                }
            }
            return league;
        }
    }
```

8. 次のコードを追加して、チームと地区の値を取得または設定するパブリッククラスを作成します。

▶ Visual Basic コードの書き方

Visual Basic

```
Public Class League
        Public Property Name() As String
            Get
                Return m_Name
            End Get
            Set(value As String)
                m_Name = Value
            End Set
        End Property
        Private m_Name As String
        Public Property Divisions() As List(Of Division)
```

```
            Get
                Return m_Divisions
            End Get
            Set(value As List(Of Division))
                m_Divisions = Value
            End Set
        End Property
        Private m_Divisions As List(Of Division)
        Public Shared Function GetLeague() As League
            Dim league = New League()
            league.Name = "主要リーグ"
            league.Divisions = New List(Of Division)()
            For Each div In "North,South,East,West".Split(","c)
                Dim d = New Division()
                league.Divisions.Add(d)
                d.Name = div
                d.Teams = New List(Of Team)()
                For Each team In "t1,t2,t3,t4".Split(","c)
                    Dim t = New Team()
                    d.Teams.Add(t)
                    t.Name = String.Format("{0} {1}", team, div)
                Next
            Next
            Return league
        End Function
    End Class
```

▶ C# コードの書き方

C#

```
public class Division
    {
        public string Name { get; set; }
        public List<Team> Teams { get; set; }
    }
    public class Team
    {
        public string Name { get; set; }
    }
}
```

9. アプリケーションを実行します。アプリケーションは次の図のようになります。

OrgChart for UWP

C1OrgChart: HierarchicalDataTemplate