OrgChart for WPF/Silverlight

2018.04.11 更新

グレープシティ株式会社

目次

製品の概要

データの構造や関係を示す階層図を作成できます。**OrgChart for WPF/Silverlight** 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。

> **メモ**:説明内に含まれるクラスおよびメンバーに対するリファレンスへのリンクは、原則としてWPF版のリファレンスページを参照します。Silverlight版については、目次から同名のメンバーを参照してください。

主な特長

OrgChart for WPF/Silverlight 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。次のような主要機能を利用して、OrgChart for WPF/Silverlight を最大限に活用してください。

- **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は、**OrgChart for WPF/Silverlight** 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 WPF/Silverlight アプリケーションを作成し、アプリケーションに **C1OrgChart** コントロールを追加します。**C1OrgChart** コントロールに表示されるランダムなデータを追加し、**OrgChart for WPF/Silverlight** で可能な実行時の操作をいくつか確認します。

手順 1:アプリケーションの作成

この手順では、**OrgChart for WPF** を使用して WPF/Silverlight 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 で新しい WPF/Silverlight プロジェクトを作成します。この例では、アプリケーションに「QuickStart」という名前を付けます。プロジェクトに別の名前を付けた場合は、後の手順で「QuickStart」を参照している箇所を実際のプロジェクトの名前に変更する必要があります。

2. 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。**[参照の追加]**ダイアログボックスで、**C1.WPF** および **C1.WPF.OrgChart** または **C1.Silverlight** および **C1.Silverlight.OrgChart** アセンブリを見つけて選択し、**[OK]**をクリックしてプロジェクトに参照を追加します。

3. MainWindow.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、いくつかのテンプレートとコントロールを追加します。

4. 次のマークアップを使用して、Window タグに XAML 名前空間を追加します。

XAML

```
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
xmlns:ext="clr-namespace:Util"
xmlns:local="clr-namespace:QuickStart"
```

プロジェクトに「QuickStart」以外の名前を指定した場合は、最後の参照をそのプロジェクト名に変更する必要があります。Window タグは次のように表示されます。

WPF

XAML

```
<Window x:Class="MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
    xmlns:ext="clr-namespace:QuickStart.Util" xmlns:local="clr-
namespace:QuickStart"
```

Silverlight

XAML

```
<UserControl x:Class="C1SilverlightCS111010.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
xmlns:ext="clr-namespace:Util" xmlns:local="clr-namespace:QuickStart"
mc:Ignorable="d" d:DesignHeight="262" d:DesignWidth="399">
```

これは、複数の名前空間を追加しなくても、ほとんどの WPF/Silverlight コントロールを使用できるようにするための統合名前空間です。

5. 次のリソースを UserControl タグの直後に追加します。XAML は次のようになります。

XAML

```
<UserControl。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" />
        <!-- 役員用のデータテンプレート -->
        <DataTemplate x:Key="_tplDirector" >
            <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>
        <!-- その他の従業員用のデータテンプレート -->
        <DataTemplate x:Key="_tplOther" >
            <Border
                Background="WhiteSmoke" BorderBrush="Black"
                BorderThickness="1 1 2 2" CornerRadius="6"
                MaxWidth="200" >
                <StackPanel Orientation="Vertical" >
                    <Border CornerRadius="6 6 0 0" Background="Black" >
                        <TextBlock Text="{Binding Name}" FontWeight="Bold"
FontSize="14"

                         Foreground="WhiteSmoke" Padding="4 0 0 0" />
                    </Border>
                    <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>
</UserControl。Resources>
```

6. Grid タグ内に次の行定義を追加します。

XAML

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

7. 行定義の直後に次のマークアップを追加して、**C1OrgChart** にタイトルを追加します。

XAML

```
<!-- サンプルタイトル -->
        <StackPanel Orientation="Horizontal" >
            <TextBlock Text="C1OrgChart クイックスタート" FontSize="16"
VerticalAlignment="Bottom" />
            <TextBlock Name="_tbTotal" VerticalAlignment="Bottom" />
        </StackPanel>
 <!-- コントロールパネル -->
<StackPanel Orientation="Horizontal" VerticalAlignment="Top" Grid。Row="1" >
    <Button Content="新しいデータ" Padding="8 0" Click="Button_Click" />
    <TextBlock Text=" ズーム:" VerticalAlignment="Center" />
    <Slider x:Name="_sliderZoom" VerticalAlignment="Center" Minimum="。01"
Maximum="1"

     Value="1" Width="200" />
</StackPanel>
<StackPanel Orientation="Horizontal" VerticalAlignment="Top" Grid。Row="2">
    <StackPanel Orientation="Horizontal" >
        <TextBlock Text="方向:" VerticalAlignment="Center" />
        <ComboBox Width="100" SelectedValue="{Binding ElementName=_orgChart,

         Path=Orientation, Mode=TwoWay}" ItemsSource=

         "{Binding Source={ext:EnumerationExtension EnumType=Orientation}}" />
    </StackPanel>
    <StackPanel Orientation="Horizontal" >
        <TextBlock Text=" 内容の横位置: " VerticalAlignment="Center" />
    <ComboBox Width="80" SelectedValue="{Binding ElementName=_orgChart,
```

```
     Path=HorizontalContentAlignment, Mode=TwoWay}"

     ItemsSource="{Binding Source={ext:EnumerationExtension
EnumType=HorizontalAlignment}}" />
    </StackPanel>
    <StackPanel Orientation="Horizontal" >
        <TextBlock Text=" 内容の縦位置： " VerticalAlignment="Center" />
        <ComboBox Width="80" SelectedValue="{Binding ElementName=_orgChart,

         Path=VerticalContentAlignment, Mode=TwoWay}"

         ItemsSource="{Binding Source={ext:EnumerationExtension
EnumType=VerticalAlignment}}" />
    </StackPanel>
</StackPanel>
```

8. 最後の **StackPanel** の直後に次の XAML マークアップを追加して、**C1OrgChart** コントロールを **ScrollViewer** に入れてアプリケーションに追加します。

XAML

```
<!-- 組織図 -->
<ScrollViewer Grid。Row="3" HorizontalScrollBarVisibility="Auto"

 VerticalScrollBarVisibility="Auto" Padding="0">
 <c1:C1OrgChart x:Name="_orgChart" Grid。Row="1" Orientation="Horizontal"

  ItemTemplateSelector="{StaticResource _personTplSelector}"

  ConnectorStroke="Black" ConnectorThickness="2" >
<!-- スケール変換をスライダに連結する -->
    <c1:C1OrgChart。RenderTransform>
        <ScaleTransform ScaleX="{Binding Value, ElementName=_sliderZoom}"

         ScaleY="{Binding Value, ElementName=_sliderZoom}" />
    </c1:C1OrgChart。RenderTransform>
</c1:C1OrgChart>
</ScrollViewer>
```

これで、「_orgChart」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。

これで、アプリケーションのユーザーインターフェイ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、コントロールに対して実行可能ないくつかの操作を確認してみます。

手順 2：コントロールへのコンテンツの追加

前の手順では、WPF/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。この手順では、いくつかのコードファイルを追加しますが、それらのファイルは **ComponentOne for WPF/Silverlight** のサンプルに付属しています。このサンプルは、デフォルトでは、**ComponentOne Samples\WPF\C1.WPF.OrgChart\OrgChart_Demo** または **ComponentOne Samples\Silverlight \C1.Silverlight.OrgChart\OrgChart_Demo** ディレクトリの **Documents** フォルダにインストールされます。コードはこのページにも記載されていますが、手っ取り早くコードファイルを見つけて追加することもできます。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、**MainWindow.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. コードファイルを次のように編集します。

VisualBasic

```
Partial Public Class MainWindow
    Inherits Window
    Public Sub New()
        InitializeComponent()
        CreateData()
    End Sub
    Private Sub Button_Click(sender As Object, e As RoutedEventArgs)
        CreateData()
    End Sub
    Private Sub CheckBox_Click(sender As Object, e As RoutedEventArgs)
        _orgChart.Orientation = If(DirectCast(sender, CheckBox).IsChecked.Value,

        Orientation.Horizontal, Orientation.Vertical)
    End Sub
    Private Sub CreateData()
        Dim p = Data.Person.CreatePerson(10)
        _tbTotal.Text = String.Format(" ({0} items total)", p.TotalCount)
        _orgChart.Header = p
    End Sub
End Class
Public Class PersonTemplateSelector
    Inherits DataTemplateSelector
    Public Overrides Function SelectTemplate(item As Object,
                        _container As DependencyObject) As DataTemplate
        Dim p = TryCast(item, Data.Person)
        Dim e = TryCast(container, FrameworkElement)
        Return If(p.Position.IndexOf("Director") > -1,
                        _TryCast(e.Resources("_tplDirector"), DataTemplate),

        TryCast(e.Resources("_tplOther"), DataTemplate))
    End Function
End Class
```

C#

```
namespace QuickStart
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
                    CreateData();        }
        void Button_Click(object sender, RoutedEventArgs e)
        {
            CreateData();
        }
        void CheckBox_Click(object sender, RoutedEventArgs e)
        {
            _orgChart.Orientation = ((CheckBox)sender).IsChecked.Value
                ? Orientation.Horizontal
                : Orientation.Vertical;
        }
        void CreateData()
        {
            var p = Data.Person.CreatePerson(10);
            _tbTotal.Text = string.Format(" ({0} items total)", p.TotalCount);
            _orgChart.Header = p;
        }
    }
    public class PersonTemplateSelector : DataTemplateSelector
    {
        public override DataTemplate SelectTemplate(object item, DependencyObject container)
        {
            var p = item as Data.Person;
            var e = container as FrameworkElement;
            return p.Position.IndexOf("Director") > -1
                ? e.Resources["_tplDirector"] as DataTemplate
                : e.Resources["_tplOther"] as DataTemplate;
        }
    }
}
```

3. ソリューションエクスプローラに移動し、プロジェクト名を右クリックし、**[追加]**→**[新しい項目]**を選択します。**[新しい項目の追加]**ダイアログボックスで、**[コードファイル]**テンプレートを見つけます。「EnumerationExtension」と名前を付け、**[追加]**をクリックします。新しいコードファイルが開きます。このコードファイルはサンプルにも含まれているので、そのファイルを単に追加することもできます。

4. 次のコードを **EnumerationExtension** コードファイルに追加します。

Visual Basic

```
Imports System.Windows
Imports System.Windows.Controls
Imports System.Windows.Markup
Imports System.Reflection
Imports System.Collections.Generic
Namespace Util
    Public Class EnumerationExtension
        Inherits MarkupExtension
        Public Property EnumType() As Type
            Get
                Return m_EnumType
            End Get
            Set(value As Type)
                m_EnumType = Value
            End Set
        End Property
        Private m_EnumType As Type
        Public Overrides Function ProvideValue(serviceProvider As IServiceProvider) As Object
            Dim list = New List(Of Object)()
            For Each value In EnumType.GetFields(BindingFlags.[Public] Or BindingFlags.[Static])
                list.Add(value.GetValue(Nothing))
            Next
            Return list
        End Function
    End Class
End Namespace
```

C#

```
using System;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Markup;
using System.Reflection;
using System.Collections.Generic;
namespace Util
{
    public class EnumerationExtension : MarkupExtension
    {
        public Type EnumType { get; set; }
        public override object ProvideValue(IServiceProvider serviceProvider)
        {
            var list = new List<object>();
            foreach (var value in EnumType.GetFields(BindingFlags.Public
            {
                list.Add(value.GetValue(null));
            }
            return list;
        }
    }
}
```

このコードは、ComboBox コントロールを列挙値に連結して、設定する連結値を実行時に選択できるようにします。

5. ソリューションエクスプローラに移動し、プロジェクト名を右クリックし、**[追加]**→**[新しい項目]**を選択します。**[新しい項目の追加]**ダイアログボックスで、**[コードファイル]**テンプレートを見つけます。「Person」と名前を付け、**[追加]**をクリックします。新しいコードファイルが開きます。このコードファイルはサンプルにも含まれているので、そのファイルを単に追加することもできます。

6. 次のコードを **Person** コードファイルに追加します。

Visual Basic

```
Imports System.Collections
Imports System.Collections.Generic
Imports System.Collections.ObjectModel
Namespace Data
    Public Class Person
```

```
        Private _list As New ObservableCollection(Of Person)()
#Region "** object model"
        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 Position() As String
            Get
                Return m_Position
            End Get
            Set(value As String)
                m_Position = Value
            End Set
        End Property
        Private m_Position As String
        Public Property Notes() As String
            Get
                Return m_Notes
            End Get
            Set(value As String)
                m_Notes = Value
            End Set
        End Property
        Private m_Notes As String
        Public ReadOnly Property Subordinates() As IList(Of Person)
            Get
                Return _list
            End Get
        End Property
        Public ReadOnly Property TotalCount() As Integer
            Get
                Dim count = 1
                For Each p In Subordinates
                    count += p.TotalCount
                Next
                Return count
            End Get
        End Property
        Public Overrides Function ToString() As String
            Return String.Format("{0}:" & vbCr & vbLf & vbTab & "{1}", Name, Position)
        End Function
#End Region
#Region "** Person factory"
        Shared _rnd As New Random()
        Shared _positions As String() = "Director|Manager|Designer|Developer|Writer|Assistant".Split("
                Shared _areas As String() = "Development|Marketing|Sales|Support|Accounting".Split("
        Shared _first As String() =

"John|Paul|Dan|Dave|Rich|Mark|Greg|Erin|Susan|Sarah|Tim|Trevor|Kevin|Mark|Dewey|Huey|Larry|Moe|Curly|Adam|Albert".Split("
        Shared _last As String() =

"Smith|Doe|Williams|Sorensen|Hansen|Mandela|Johnson|Ward|Woodman|Jordan|Mays|Kevorkian|Trudeau|Hendrix|Clinton".Split("
        Shared _verb As String() =

"likes|reads|studies|hates|exercises|dreams|plays|writes|argues|sleeps|ignores".Split("
        Shared _adjective As String() =

"long|short|important|pompous|hard|complex|advanced|modern|boring|strange|curious|obsolete|bizarre".Split("
        Shared _noun As String() =

"products|tasks|goals|campaigns|books|computers|people|meetings|food|jokes|accomplishments|screens|pages".Split("
        Public Shared Function CreatePerson(level As Integer) As Person
            Dim p = CreatePerson
                If level > 0 Then
                level -= 1
                For i As Integer = 0 To _rnd.[Next](1, 4) - 1
                    p.Subordinates.Add(CreatePerson(_rnd.[Next](level \ 2, level)))
                Next
            End If
            Return p
        End Function
        Public Shared Function CreatePerson() As Person
```

```
            Dim 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() < 0.5
                p.Notes += String.Format(" and {0} {1} {2}", GetItem(_verb), GetItem(_adjective), GetItem(_noun))
            End While
            p.Notes += "."
            Return p
        End Function
        Private Shared Function GetItem(list As String()) As String
            Return list(_rnd.[Next](0, list.Length))
        End Function
#End Region
    End Class
End Namespace
```

C#

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Collections.ObjectModel;
namespace Data
{
    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 = "Director|Manager|Designer|Developer|Writer|Assistant".Split('
        static string[] _areas = "Development|Marketing|Sales|Support|Accounting".Split('
        static string[] _first =

"John|Paul|Dan|Dave|Rich|Mark|Greg|Erin|Susan|Sarah|Tim|Trevor|Kevin|Mark|Dewey|Huey|Larry|Moe|Curly|Adam|Albert".Split('
        static string[] _last =

"Smith|Doe|Williams|Sorensen|Hansen|Mandela|Johnson|Ward|Woodman|Jordan|Mays|Kevorkian|Trudeau|Hendrix|Clinton".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
    }
}
```

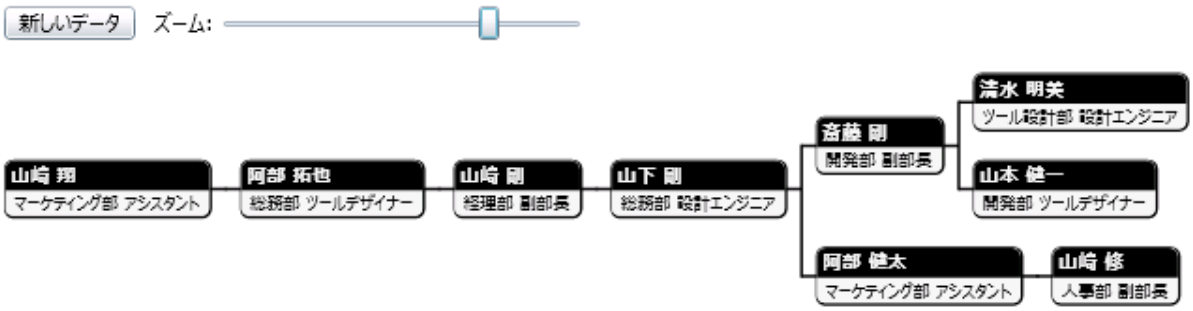
このコードは、**C1OrgChart** に項目を表示するための再帰的な **ObservableCollection** を含む単純なクラスを作成し、単純な組織図を構築するためのメソッドを作成します。**C1OrgChart** コントロールのデフォルトのテンプレートは、**ToString** メソッドを使用して、表示するテキストを決定します。**DataTemplate** を使用しない場合は、(上のコードのように)このメソッドをオーバーライドする必要があります(**DataTemplate** を使用する場合は、そこで連結を設定し、**ToString** は無視できます)。

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、このコントロールで可能な実行時の操作をいくつか示します。

手順 3:アプリケーションの実行

WPF 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。アプリケーションを実行して OrgChart for WPF/Silverlight 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。アプリケーションは次の図のように表示されます。



   このアプリケーションには、コントロールパネル領域とランダムな組織図が含まれます。

2. スライダのサムボタンをドラッグして、**C1OrgChart** のズームレベルを設定します。
3. **[方向]**ドロップダウンボックスをクリックし、**[Vertical]**を選択して、コントロールの方向を変更します。グリッドが垂直方向に表示されます。
4. **[内容の横位置]**および**[内容の縦位置]**ドロップダウンボックスのオプションを選択して、コントロールの配置を設定します。
5. **[新しいデータ]**ボタンをクリックして、コントロールに別のデータをロードします。

おめでとうございます!

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

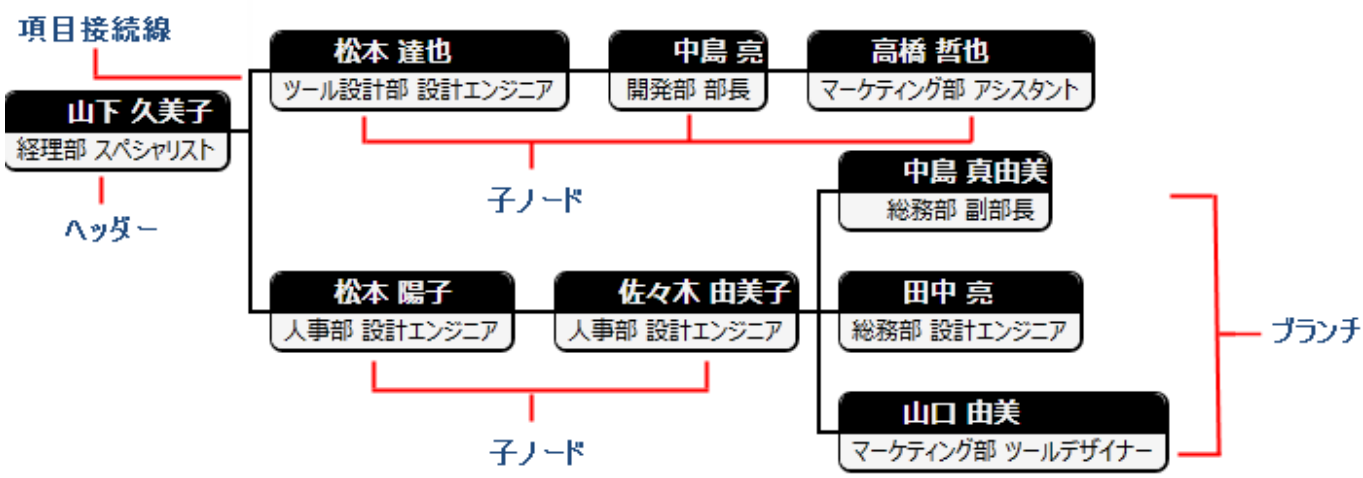
OrgChart for WPF の使い方

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 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。

OrgChart 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、**Employee** クラスがあるとすると、このクラスは、この従業員に関するいくつかのプロパティと、この親 **Employee** 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。

```
public class Employee
{
  List<Employee> _list = new List<Employee>();
  public string Name { get; set; }
  public string Position { get; set; }
  public string Notes { get; set; }
  public IEnumerable<Employee> Subordinates{ get; set;

}
```

1つの **Employee** オブジェクトを **Header** プロパティに割り当てると、**C1OrgChart** は、**Subordinates** プロパティに **Employee** オブジェクトのコレクションが含まれていることを自動的に検出します。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 プロパティは、**C1OrgChart** コントロールにデータ項目を表示する方法を指定します。これは WPF/Sliverlight の標準 **DataTemplate** で、XAML で次のように定義できます。

XAML

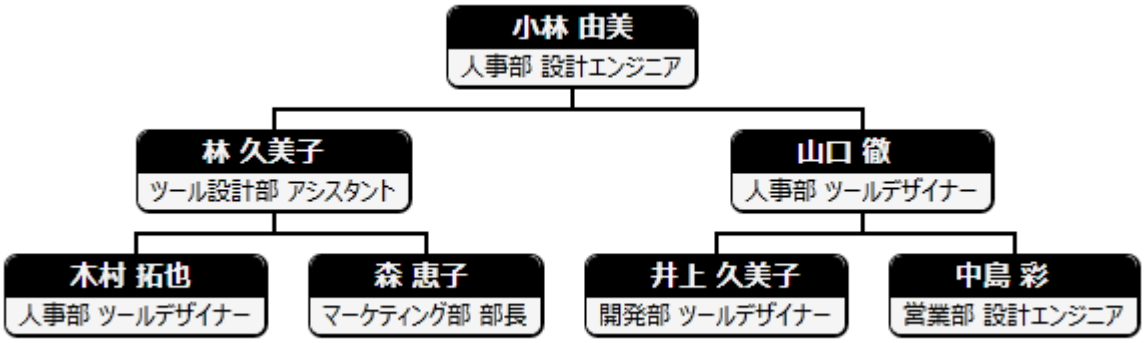
```
<Window.Resources>
   <!-- C1OrgChart ノードコンテンツ -->
   <DataTemplate x:Key="EmployeeTemplate" >
     <Border
       Background="WhiteSmoke" BorderBrush="Black"
       BorderThickness="1 1 2 2" CornerRadius="6"
       MaxWidth="200" >
       <StackPanel Orientation="Vertical" >
         <TextBlock Text="{Binding Name}" FontSize="14" />
         <TextBlock Text="{Binding Notes}" FontSize="9.5" />
         <TextBlock Text="{Binding Position}" FontSize="12" />
       </StackPanel>
     </Border>
  </DataTemplate>
 </Window.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。

XAML

```
<c1:C1OrgChart
   x:Name="_orgChart"
   ItemTemplate="{StaticResource EmployeeTemplate }" >
 </c1:C1OrgChart>
```

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。

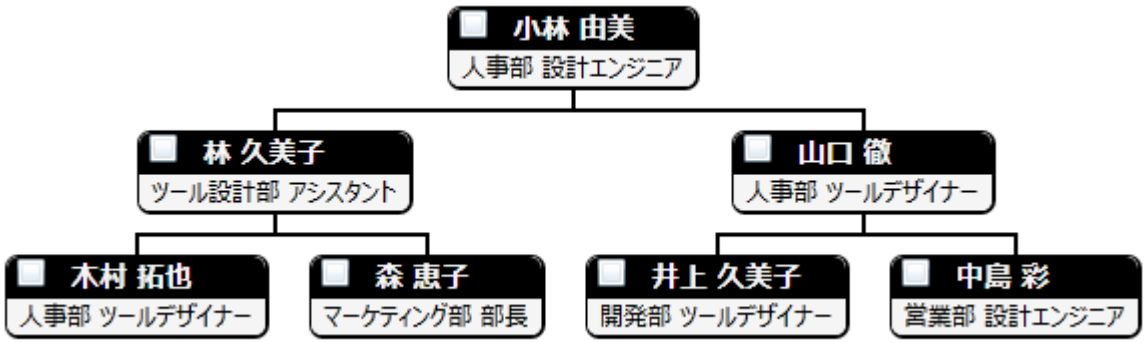
このような例として、**CheckBox。IsChecked** プロパティを **C1OrgChart** の **IsCollapsed** プロパティに連結するとたいへん便利です。これにより、**TreeView** のように動作する折りたたみ可能な **C1OrgChart** を作成できます。

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を多少変更したバージョンです。

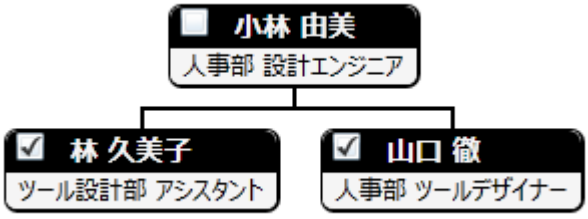
XAML

```
<Window.Resources>
<!-- C1OrgChart ノードコンテンツ -->
<DataTemplate x:Key="EmployeeTemplate" >
<Border
Background="WhiteSmoke" BorderBrush="Black"
BorderThickness="1 1 2 2" CornerRadius="6"
MaxWidth="200" >
<StackPanel Orientation="Vertical" >
<!-- CheckBox を C1OrgChart IsCollapsed プロパティに連結する -->
<CheckBox Margin="4 0"
IsChecked="{Binding IsCollapsed, Mode=TwoWay,
RelativeSource={RelativeSource AncestorType=c1:C1OrgChart}}"/>
<TextBlock Text="{Binding Name}" FontSize="14" />
<TextBlock Text="{Binding Notes}" FontSize="9。5" />
<TextBlock Text="{Binding Position}" FontSize="12" />
</StackPanel>
</Border>
</DataTemplate>
</Window.Resources>
```

この変更により、組織図は次のようになります。



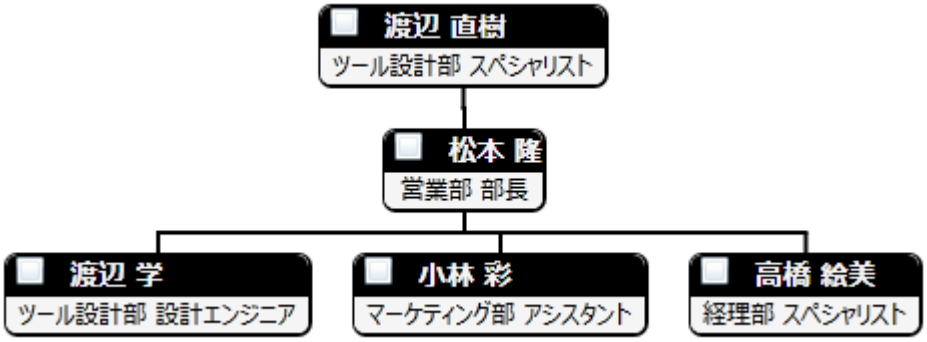
チェックボックスをクリックすると、ブランチが折りたたまれ、コンパクトに表示されます。



また、連結を使用して、接続線をカスタマイズすることもできます。たとえば、次の XAML は、子ノードの数に応じて接続線の太さが変わる組織図を生成します。

XAML

```
<c1:C1OrgChart
x:Name="_orgChart" >
ConnectorThickness="{Binding Path=Subordinates.Count}"
ItemTemplate="{StaticResource EmployeeTemplate }" >
<c1:C1OrgChart>
```



John Doe には直属の部下が1人しかいないため、接続線は1ピクセルの太さになります。Kevin Smith には直属の部下が3人いるため、接続線は3ピクセルの太さになります。

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：このクラスを使用すると、特定のデータ項目の特性に基づいて、異なるテンプレートを選択できます。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#

```
/// <summary>
/// 作成される項目のテンプレートを動的に選択するために使用されるクラス。
/// </summary>
public class EmployeeTemplateSelector : DataTemplateSelector
{
  public override DataTemplate SelectTemplate(object item, DependencyObject ctnr)
  {
    var p = item as Employee;
    var e = Application.Current.RootVisual as FrameworkElement;
    return p.Position.IndexOf("Director") > -1
      ? e.Resources["DirectorTemplate"] as DataTemplate
      : e.Resources["EmployeeTemplate"] as DataTemplate;
  }
}
```

次の点が重要です。

- WPF では、**DataTemplateSelector** クラスは System.Windows.Controls 名前空間に定義されています。

- **DataTemplateSelector** クラスは、**C1OrgChart** コントロールに固有ではありません。これは、任意の **ItemsControl** オブジェクト（**ListBox.ItemTemplateSelector** など）と共に使用される標準の WPF クラスです。
- このコードは、アプリケーションのルートビジュアルに "DirectorTemplate" および "EmployeeTemplate" データテンプレートがリソースとして定義されているとしています。

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

```
<Window.Resources>
   <!-- 従業員の役職に基づくテンプレートの選択 -->
   <local:PersonTemplateSelector x:Key="TemplateSelector" />
   <!-- 役員用のデータテンプレート -->
   <DataTemplate x:Key="DirectorTemplate" >
   …
  </DataTemplate>
   <!-- 他の従業員用のデータテンプレート -->
   <DataTemplate x:Key="EmployeeTemplate" >
   …
  </DataTemplate>
 </Window.Resources>
   …
  <c1:C1OrgChart
     ItemTemplateSelector="{StaticResource TemplateSelector}"
     ConnectorStroke="Black" ConnectorThickness="3" />
```

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

```
<Window.Resources>
   <!-- Team オブジェクト用の通常のテンプレート -->
   <DataTemplate x:Key="TeamTemplate" >
     <Border Background="LightBlue" Padding="4" >
       <TextBlock FontStyle="Italic" Text="{Binding Path=Name}" />
     </Border>
   </DataTemplate>
   <!-- Division オブジェクト用の階層化テンプレート -->
   <c1:HierarchicalDataTemplate x:Key="DivisionTemplate"
     ItemsSource="{Binding Path=Teams}"
     ItemTemplate="{StaticResource TeamTemplate}">
    <Border Background="Gold" >
       <TextBlock Text="{Binding Path=Name}" FontWeight="Bold" Padding="20" />
     </Border>
   </c1:HierarchicalDataTemplate>
   <!-- League オブジェクト用の階層化テンプレート -->
   <c1:HierarchicalDataTemplate x:Key="LeagueTemplate"
     ItemsSource="{Binding Path=Divisions}"
     ItemTemplate="{StaticResource DivisionTemplate}">
     <Border Background="LightCoral" >
       <TextBlock Text="{Binding Path=Name}" FontWeight="Bold" Padding="40" />
     </Border>
   </c1:HierarchicalDataTemplate>
 </Window.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。

- WPF では、**HierarchicalDataTemplate** クラスは System.Windows 名前空間に定義されています。
- **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。

<c1:C1OrgChart ItemTemplate="{StaticResource LeagueTemplate}" >

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。これは、**Menu** や **TreeView** などの WPF **HeaderedItemControl** クラスで使用されているメカニズムと同じです。

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。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、グリッドの外観をカスタマイズしたり、WPF の XAML ベースのスタイル設定を活用することができます。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。

パネル内のレイアウト

WPF/Silverlight 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** パネルでは **Row**、**ColumnSpan**、および **RowSpan** プロパティ、**Canvas** パネルでは **Left** および **Top** 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。

外観プロパティ

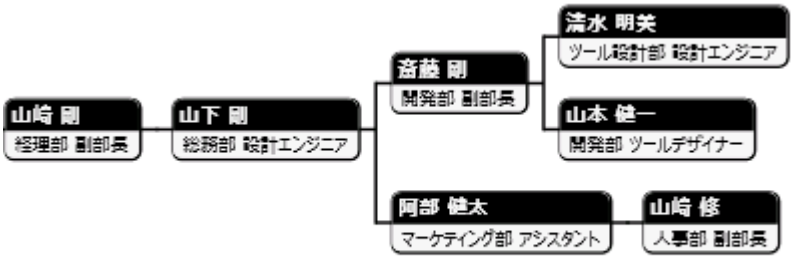
ItemTemplate プロパティを使用して、**C1OrgChart** のノードの外観を完全に自由に指定できます。**C1OrgChart** は、組織図自体の外観をカスタマイズするためのプロパティをいくつか公開します。

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。

XAML

```
<c1:C1OrgChart
  x:Name="_orgChart" Orientation="Horizontal"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
<c1: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。

XAML

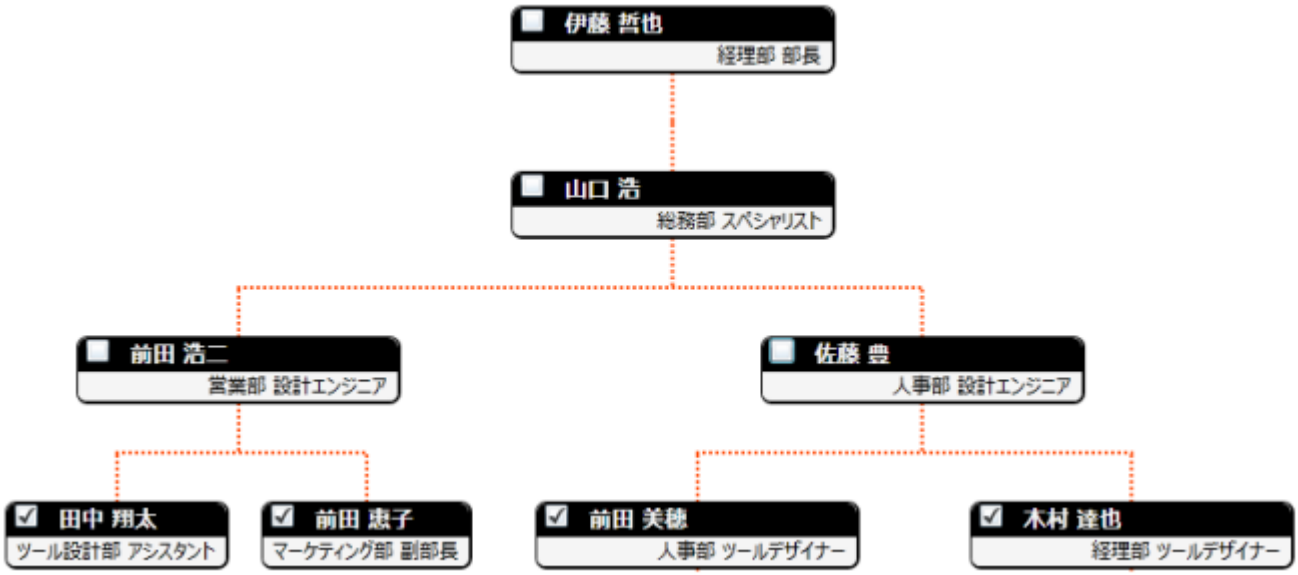
```
<c1:C1OrgChart
x:Name="_orgChart" FlowDirection="RightToLeft"
ItemTemplate="{StaticResource EmployeeTemplate }" >
<c1:C1OrgChart>
```

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。

XAML

```
<c1:C1OrgChart
  x:Name="_orgChart"
  ChildSpacing="20 60"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
<c1:C1OrgChart>
```



デフォルトの設定と比べて、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。

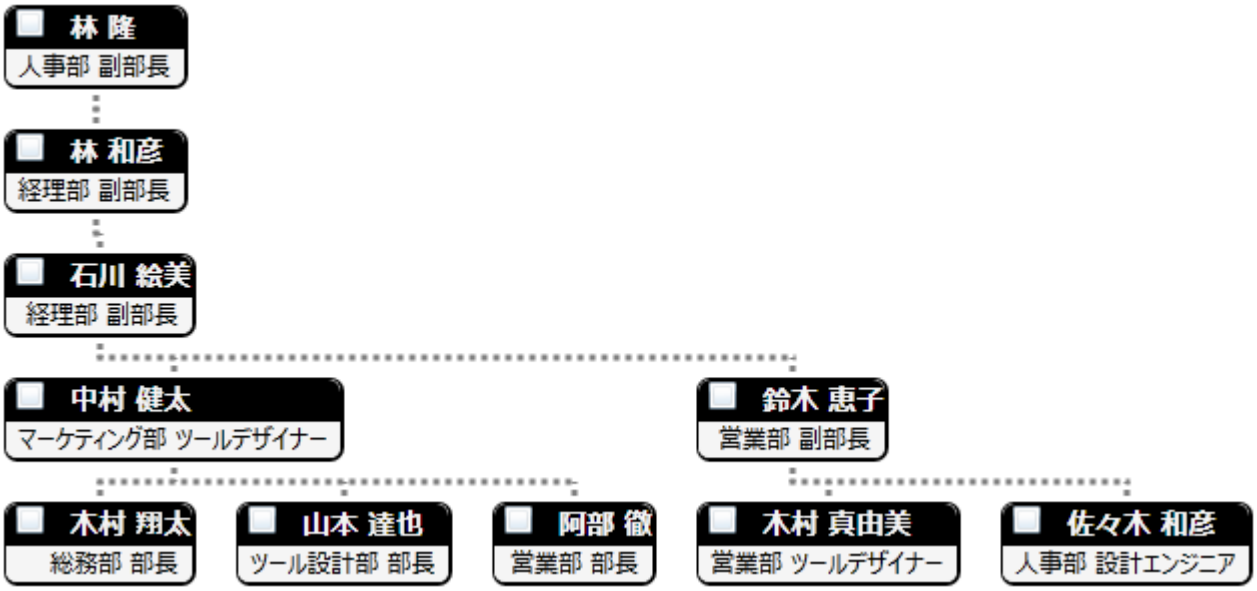
Connector

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 要素のプロパティと似ています。

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。

XAML

```
<c1:C1OrgChart
  x:Name="_orgChart"
  ConnectorStroke="Gray"
  ConnectorThickness="3"
  ConnectorDashArray="1 1"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
<c1:C1OrgChart>
```



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：

XAML

```
<c1:C1OrgChart
  x:Name="_orgChart"
  HorizontalContentAlignment="Left"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
<c1:C1OrgChart>
```

OrgChart は次のように表示されます。

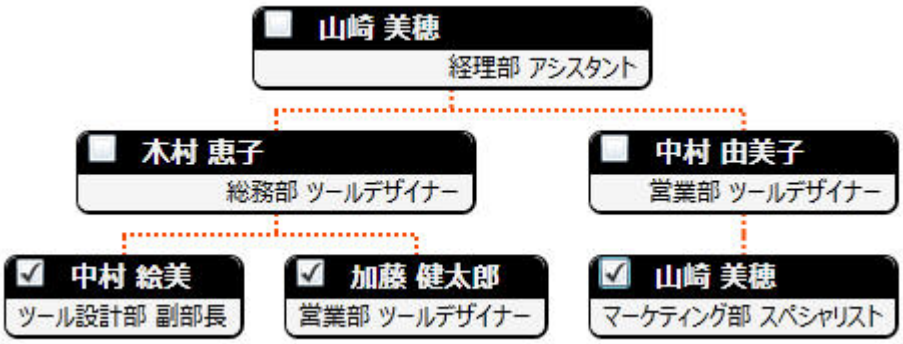
Stretch

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

XAML

```
<c1:C1OrgChart
  x:Name="_orgChart"
  HorizontalContentAlignment="Stretch"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
<c1:C1OrgChart>
```

OrgChart は次のように表示されます。



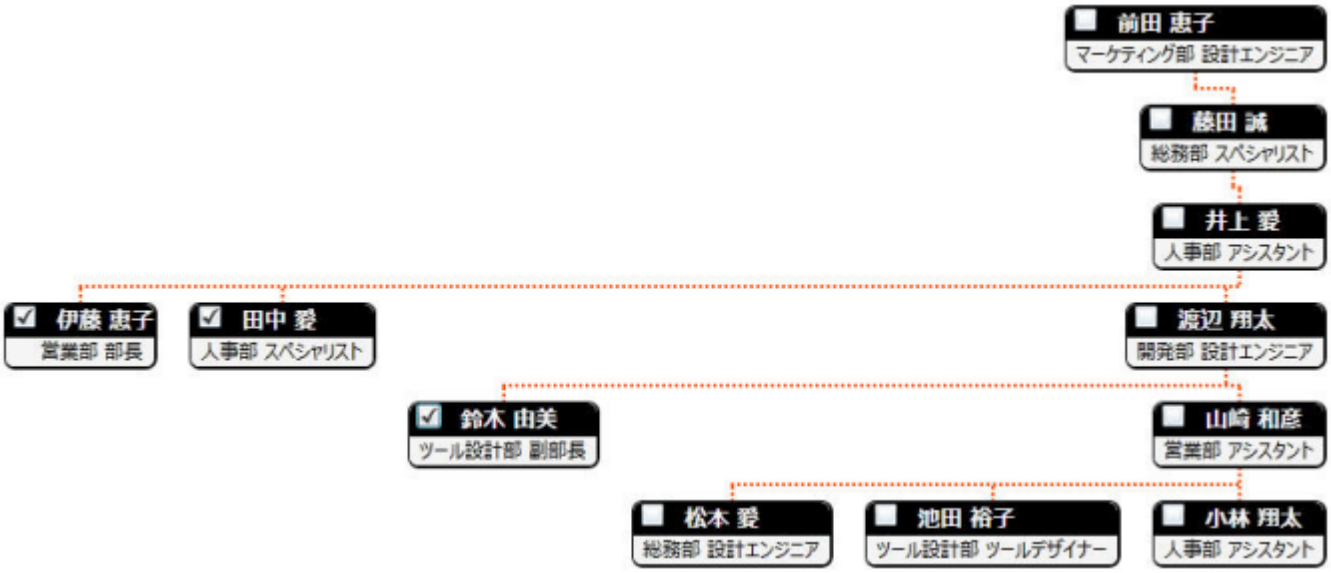
Right

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

XAML

```
<c1:C1OrgChart
  x:Name="_orgChart"
  HorizontalContentAlignment="Right"
  ItemTemplate="{StaticResource EmployeeTemplate }" >
</c1:C1OrgChart>
```

OrgChart は次のように表示されます。



テンプレート

WPF/Silverlight コントロールを使用する主な利点の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ーフェイスを持つ「外観のない」コントロールであることです。WPF/Silverlight アプリケーションのユーザーインターフェイスであるルックアンドフィールを独自に設計するのと同様に、**OrgChart for WPF/Silverlight** で管理されるデータに関して独自の 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、**C1OrgChart** 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。**[コピーして編集]**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、**[空アイテムの作成]**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**:メニューを使用して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 `Template` 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、**C1OrgChart** 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 **C1OrgChart** 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、**[空アイテムの作成]**を選択します。

表示状態

Microsoft Expression Blend で、カスタム状態や状態グループを追加して、ユーザーコントロールの状態ごとに異なる外観を定義できます。たとえば、マウスが置かれたときのコントロールの表示状態を変更できます。新しいテンプレートを作成し、新

しいテンプレートパーツを追加することで、表示状態を表示および編集できます。これで、そのパーツで利用可能な表示状態が**[表示状態]**ウィンドウに表示されます。

よく使用される状態としては、項目の通常の外観を示す Normal、マウスが置かれている項目を示す MouseOver、有効でない項目を示す Disabled などがあります。フォーカスの状態には、項目にフォーカスがないときの Unfocused、項目にフォーカスがあるときの Focused などがあります。

タスク別ヘルプ

次のタスク別ヘルプのトピックは、ユーザーの皆様が Visual Studio および Expression Blend 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 WPF/Silverlight** 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 WPF/Silverlight** 製品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。タスク別ヘルプの多くのトピックは、新しい WPF/Silverlight プロジェクトが作成されており、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の詳細については、「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 の**[ファイル]**メニューから、**[新規作成]**を選択し、**[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、左側のメニューから言語を選択し、**[フレームワーク]**ドロップダウンリストで**[.NET Framework 4]**を選択し、プロジェクトの名前を入力します。
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。**[参照の追加]**ダイアログボックスで、以下のアセンブリを見つけて選択し、**[OK]**をクリックしてプロジェクトに参照を追加します。
   - C1.WPF および C1.WPF.OrgChart
   - C1.Silverlight および C1.Silverlight.OrgChart
4. MainWindow.xaml ファイルの XML ビューを開き、マークアップ xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml" を使用して、Window タグに XAML 名前空間を追加します。名前空間は次のようになります。

   XAML
   ```
   <Window x:Class="MainWindow"
       xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
       xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
       xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
       Title="MainWindow" Height="350" Width="525">
   ```

   これは、複数の名前空間を追加しなくても、ほとんどの WPF コントロールを使用できるようにするための統合名前空間です。
5. ページの Grid タグ内に <c1:C1OrgChart x: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。

   XAML
   ```
   <Grid x:Name="LayoutRoot" Background="White">
           <c1:C1OrgChart x: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。

これでアプリケーションは正しくセットアップされましたが、ここでアプリケーションを実行しても、アプリケーションは空のままです。

メモ: C1OrgChart コントロールが VisualStudio のツールボックスにインストールされている場合は、ページにコントロールをドラッグするだけで上のすべての手順が自動的に実行されます。

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

<c1:C1OrgChart> 開始タグに、**Orientation=Vertical** を挿入します。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。

```
XAML
<c1:C1OrgChart x: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。

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

<c1:C1OrgChart> 開始タグを見つけ、そのタグに **FlowDirection="RightToLeft"** を挿入します。<c1:C1OrgChart> マークアップは次のようになります。

```
XAML
<c1:C1OrgChart x: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。

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され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 <c1: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。

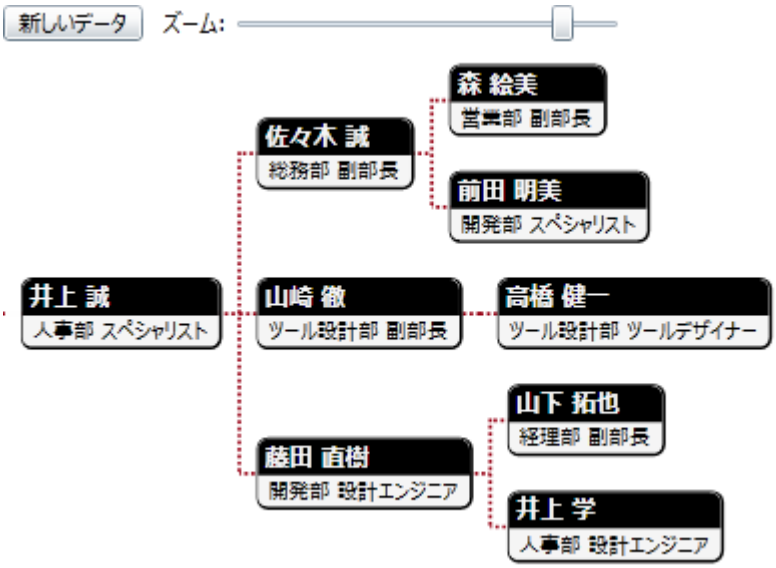
最終的な XAML マークアップは次のようになります。

```
Example Title
<c1:C1OrgChart x: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。



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. Visual Studio の**[ファイル]**メニューから、**[新規作成]**を選択し、**[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、左側のメニューから言語を選択します。**[フレームワーク]**ドロップダウンリストで**[。NET Framework 4]**を選択し、プロジェクトの名前として**「OrgChart」**と入力します。
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。**[参照の追加]**ダイアログボックスで、以下のアセンブリを見つけて選択し、**[OK]**をクリックしてプロジェクトに参照を追加します。
   - C1.WPF および C1.WPF.OrgChart
   - C1.Silverlight および C1.Silverlight.OrgChart
4. xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml" 名前空間を <Window> タグの名前空間宣言に追加します。これは、ほとんどの WPF コントロールで機能する一般的な名前空間です。
5. 次の名前空間をアプリケーションの <Window> タグに追加します。

   xmlns:local="clr-namespace:OrgChart"
6. 次の XAML マークアップを <c1:C1OrgChart> </c1:C1OrgChart> タグの直前に挿入して、**C1OrgChart** データテンプレートを作成します。

XAML

```
<Window.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>
                    <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">
                                    <CheckBox Margin="4 0" IsChecked="{Binding IsCollapsed, Mode=TwoWay,
```

```
                        RelativeSource={RelativeSource AncestorType=c1:C1OrgChart}}"/>
                                <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.OtherTemplate>
            <!--その他の従業員用のデータテンプレート -->
            <DataTemplate>
                <Border Background="WhiteSmoke" BorderBrush="Black" BorderThickness="1 1 2 2"
CornerRadius="6" MaxWidth="200">
                    <StackPanel Orientation="Vertical">
                        <Border CornerRadius="6 6 0 0" Background="Black">
                            <StackPanel Orientation="Horizontal">
                                <CheckBox Margin="4 0" IsChecked="Binding IsCollapsed, Mode=TwoWay,

                        RelativeSource={RelativeSource AncestorType=c1:C1OrgChart}}"/>
                                <TextBlock Text="{Binding Name}" FontWeight="Bold" FontSize="14"

                        foreground="WhiteSmoke" Padding="4 0 0 0"></TextBlock>
                            </StackPanel>
                        </Border>
                        <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>
        </local:PersonTemplateSelector.OtherTemplate>
    </local:PersonTemplateSelector>
</Window.Resources>
```

7. 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

```
<!--組織図 -->
    <ScrollViewer Background="White" Grid.Row="1" HorizontalScrollBarVisibility="Auto"

     VerticalScrollBarVisibility="Auto" Padding="0">
        <c1:C1OrgChart x:Name="_orgChart" Grid.Row="1" Orientation="Horizontal"

         ItemTemplateSelector="{StaticResource _personTplSelector}"
                      ConnectorStroke="Black" ConnectorThickness="2" IsCollapsed="False">
            <!--スケール変換をスライダに連結する -->
            <c1:C1OrgChart.RenderTransform>
                <ScaleTransform ScaleX="{Binding Value, ElementName=_sliderZoom}"

                 ScaleY="{Binding Value, ElementName=_sliderZoom}" />
            </c1:C1OrgChart.RenderTransform>
            <!--ツリーノードの表示に使用されるテンプレート -->
            <!--テンプレートセレクタを使用するため、このサンプルでは使用しません -->
            <!--<c1:C1OrgChart.ItemTemplate />-->
        </c1:C1OrgChart>
    </ScrollViewer>
```

8. 次の XAML マークアップを </Window.Resources> タグと <c1:C1OrgChart> タグの間に追加します。

XAML

```
<!--レイアウトルート -->
    <Grid x:Name="LayoutRoot">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition />
        </Grid.RowDefinitions>
       <!--コントロールパネル -->
        <StackPanel Orientation="Horizontal" VerticalAlignment="Top" Margin="0 8">
            <Button Content="新しいデータ" Padding="8 0" Click="Button_Click" />
            <TextBlock Text="  ズーム: " VerticalAlignment="Center" />
```

```
            <Slider x:Name="_sliderZoom" VerticalAlignment="Center" Minimum=".01" Maximum="1" Value="1" Width="200" />
        </StackPanel>
```

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。次の名前空間をコードファイルにインポートします。

WPF

VisualBasic

```
Imports C1.WPF.OrgChart
```

C#

```
using C1.WPF.OrgChart;
```

Silverlight

VisualBasic

```
Imports C1.Silverlight.OrgChart
```

C#

```
using C1.Silverlight.OrgChart;
```

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

Visual Basic

```
CreateData()
    End Sub
    Private Sub Button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
        CreateData()
    End Sub
    Private Sub CreateData()
        Dim p = Data.Person.CreatePerson(10)
        _orgChart.Header = p
    End Sub
    Public Property DEMO_Orientation As Orientation
        Get
            Return _orgChart.Orientation
        End Get
        Set(value As Orientation)
            _orgChart.Orientation = value
        End Set
    End Property
    Public Property DEMO_HorizontalContentAlignment As HorizontalAlignment
        Get
            Return _orgChart.HorizontalContentAlignment
        End Get
        Set(value As HorizontalAlignment)
            _orgChart.HorizontalContentAlignment = value
        End Set
    End Property
    Public Property DEMO_VerticalContentAlignment As VerticalAlignment
        Get
            Return _orgChart.VerticalContentAlignment
        End Get
        Set(value As VerticalAlignment)
            _orgChart.VerticalContentAlignment = value
        End Set
            End Property
End Class
```

C#

```
CreateData();
        }
        void Button_Click(object sender, RoutedEventArgs e)
        {
            CreateData();
        }
        void CreateData()
        {
            var p = Data.Person.CreatePerson(10);
```

```
            _orgChart.Header = p;
        }
        public Orientation DEMO_Orientation
        {
               get
            {
                return _orgChart.Orientation;
            }
            set
            {
                _orgChart.Orientation = value;
            }
        }
        public HorizontalAlignment DEMO_HorizontalContentAlignment
        {
            get
            {
                return _orgChart.HorizontalContentAlignment;
            }
            set
            {
                _orgChart.HorizontalContentAlignment = value;
            }
        }
        public VerticalAlignment DEMO_VerticalContentAlignment
        {
            get
            {
                return _orgChart.VerticalContentAlignment;
            }
            set
            {
                _orgChart.VerticalContentAlignment = value;
            }
        }
    }
```

11. 次のコードを追加して、作成される項目のテンプレートを選択します。

Visual Basic

```
' 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。
Public Class PersonTemplateSelector
    Inherits DataTemplateSelector
    Public Overrides Function SelectTemplate(item As Object, container As DependencyObject) As DataTemplate
        Dim p = TryCast(item, Data.Person)
        'var e = Application.Current.RootVisual as FrameworkElement;
        'return p.Position.IndexOf("Director") > -1
        '    ? e.Resources["_tplDirector"] as DataTemplate
        '    : e.Resources["_tplOther"] as DataTemplate;
        Return If(p.Position.IndexOf("Director") > -1, DirectorTemplate, OtherTemplate)
    End Function
    ' 組織図を指定されたレベルまで折りたたみます
    Private Sub CollapseExpand(node As C1.WPF.OrgChart.C1OrgChart, level As Integer, maxLevel As Integer)
        If level >= maxLevel Then
            node.IsCollapsed = True
        Else
            node.IsCollapsed = False
            For Each subNode In node.ChildNodes
                CollapseExpand(subNode, level + 1, maxLevel)
            Next
        End If
    End Sub
    Public Property DirectorTemplate() As DataTemplate
        Get
            Return m_DirectorTemplate
        End Get
        Set(value As DataTemplate)
            m_DirectorTemplate = Value
        End Set
    End Property
    Private m_DirectorTemplate As DataTemplate
    Public Property OtherTemplate() As DataTemplate
        Get
            Return m_OtherTemplate
        End Get
```

```
        Set(value As DataTemplate)
            m_OtherTemplate = Value
        End Set
    End Property
    Private m_OtherTemplate As DataTemplate
End Class
```

C#

```
/// 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 : DataTemplateSelector
    {
        public override DataTemplate SelectTemplate(object item, DependencyObject container)
        {
            var p = item as Data.Person;
            //var e = Application.Current.RootVisual as FrameworkElement;
            //return p.Position.IndexOf("Director") > -1
            //    ? e.Resources["_tplDirector"] as DataTemplate
            //    : e.Resources["_tplOther"] as DataTemplate;
            return p.Position.IndexOf("Director") > -1
                ? DirectorTemplate
                : OtherTemplate;
        }
        // 組織図を指定されたレベルまで折りたたみます
        void CollapseExpand(C1OrgChart node, int level, int maxLevel)
        {
            if (level >= maxLevel)
            {
                node.IsCollapsed = true;
            }
            else
            {
                node.IsCollapsed = false;
                foreach (var subNode in node.ChildNodes)
                {
                    CollapseExpand(subNode, level + 1, maxLevel);
                }
            }
        }
        public DataTemplate DirectorTemplate { get; set; }
        public DataTemplate OtherTemplate { get; set; }
    }
}
```

12. 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。
13. 次の名前空間を **Person** コードファイルに追加します。

Visual Basic

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

C#

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

14. 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

```
'''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。
Public Class PersonTemplateSelector
    Inherits DataTemplateSelector
    Public Overrides Function SelectTemplate(item As Object, container As DependencyObject) As DataTemplate
```

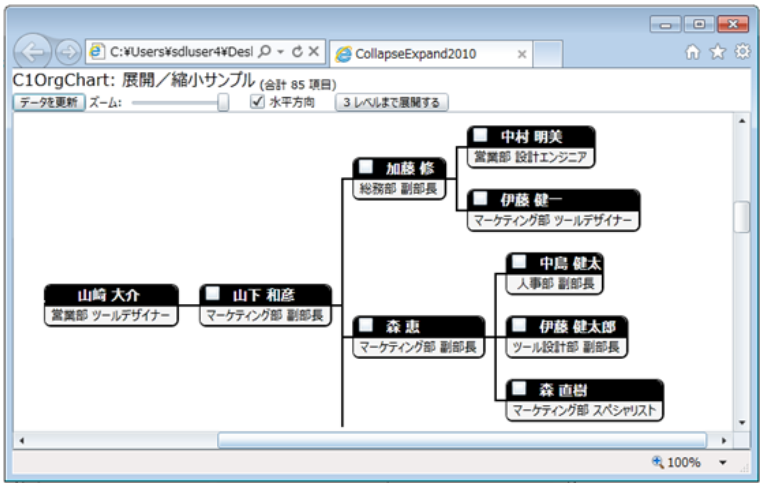
```
        Dim p = TryCast(item, Data.Person)
        'var e = Application.Current.RootVisual as FrameworkElement;
        'return p.Position.IndexOf("Director") > -1
        '    ? e.Resources["_tplDirector"] as DataTemplate
        '    : e.Resources["_tplOther"] as DataTemplate;
        Return If(p.Position.IndexOf("Director") > -1, DirectorTemplate, OtherTemplate)
    End Function
    ' 組織図を指定されたレベルまで折りたたみます
    Private Sub CollapseExpand(node As C1.WPF.OrgChart.C1OrgChart, level As Integer, maxLevel As Integer)
        If level >= maxLevel Then
            node.IsCollapsed = True
        Else
            node.IsCollapsed = False
            For Each subNode In node.ChildNodes
                CollapseExpand(subNode, level + 1, maxLevel)
            Next
        End If
    End Sub
    Public Property DirectorTemplate() As DataTemplate
        Get
            Return m_DirectorTemplate
        End Get
        Set(value As DataTemplate)
            m_DirectorTemplate = Value
        End Set
    End Property
    Private m_DirectorTemplate As DataTemplate
    Public Property OtherTemplate() As DataTemplate
        Get
            Return m_OtherTemplate
        End Get
        Set(value As DataTemplate)
            m_OtherTemplate = Value
        End Set
    End Property
    Private m_OtherTemplate As DataTemplate
End Class
```

C#

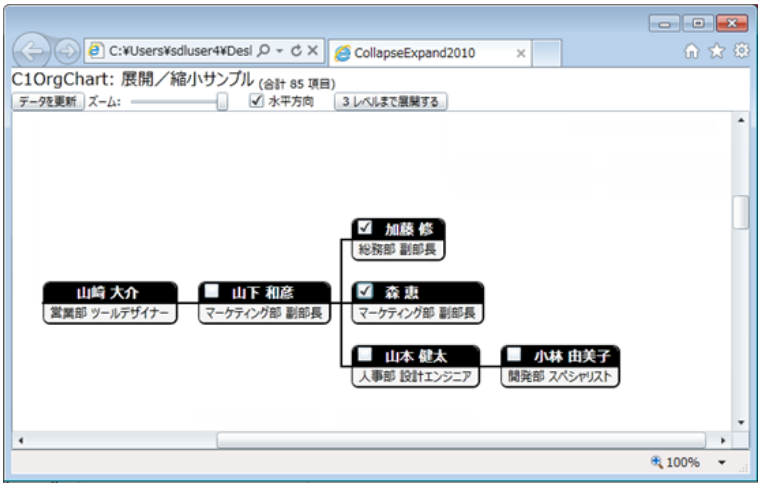
```
namespace Data
{
    /// <summary>
    /// 階層化データ項目:Person は型 Person の Subordinates を持つ。
    /// </summary>
    public class Person
    {
        ObservableCollection<Person> _list = new ObservableCollection<Person>();
        #region ** object model
        public string Name { get; set; }
        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 = "Director|Manager|Designer|Developer|Writer|Assistant".Split('
```

```
        static string[] _areas = "Development|Marketing|Sales|Support|Accounting".Split('
        static string[] _first =

"John|Paul|Dan|Dave|Rich|Mark|Greg|Erin|Susan|Sarah|Tim|Trevor|Kevin|Mark|Dewey|Huey|Larry|Moe|Curly|Adam|Albert".Split('
        static string[] _last =

"Smith|Doe|Williams|Sorensen|Hansen|Mandela|Johnson|Ward|Woodman|Jordan|Mays|Kevorkian|Trudeau|Hendrix|Clinton".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
    }
}
```

15. **[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。

1. Visual Studio の**[ファイル]**メニューから、**[新規作成]**を選択し、**[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、左側のメニューから言語を選択します。**[フレームワーク]**ドロップダウンリストで**[.NET Framework 4]**を選択し、プロジェクトの名前として**「OrgChart」**と入力します。
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。**[参照の追加]**ダイアログボックスで、以下のアセンブリを見つけて選択し、**[OK]**をクリックしてプロジェクトに参照を追加します。
   - C1.WPF および C1.WPF.OrgChart
   - C1.Silverlight および C1.Silverlight.OrgChart
4. xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml" 名前空間を <Window> タグの名前空間宣言に追加します。これは、ほとんどの ComponentOne WPF コントロールで機能する一般的な名前空間です。
5. 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

```
<UserControl.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 Path=Name}" />
            </Border>
        </DataTemplate>
        <!-- Division オブジェクト用のテンプレート -->
        <sdk:HierarchicalDataTemplate x:Key="DivisionTemplate"
            ItemsSource="{Binding Path=Teams}"
            ItemTemplate="{StaticResource TeamTemplate}">
            <Border Background="Gold" >
                <TextBlock Text="{Binding Path=Name}" FontWeight="Bold"
HorizontalAlignment="Center"

                 VerticalAlignment="Center" Padding="20" />
            </Border>
```

```
        </sdk:HierarchicalDataTemplate>
        <!-- League オブジェクト用のテンプレート -->
        <sdk:HierarchicalDataTemplate x:Key="LeagueTemplate"
            ItemsSource="{Binding Path=Divisions}"
            ItemTemplate="{StaticResource DivisionTemplate}">
            <Border Background="LightCoral" >
                <TextBlock Text="{Binding Path=Name}" FontWeight="Bold"
HorizontalAlignment="Center"

                 VerticalAlignment="Center" Padding="40" />
            </Border>
        </sdk:HierarchicalDataTemplate>
</UserControl.Resources>
```

6. 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

```
<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 Name="_tbTotal" VerticalAlignment="Bottom" />
        </StackPanel>
        <ScrollViewer Grid.Row="1" HorizontalScrollBarVisibility="Auto"
VerticalScrollBarVisibility="Auto" Padding="0" >
            <c1:C1OrgChart
                Name="_chart" ItemTemplate="{StaticResource LeagueTemplate}"
                ConnectorDashArray="1 2" ConnectorStroke="Gray"
                HorizontalAlignment="Center" VerticalAlignment="Center" />
        </ScrollViewer>
```

7. ツールボックスで通常の **TreeView** 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。<sdk:TreeView> タグに次のコードを挿入します。

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

8. 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。

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

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#

```
//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 };
        }
    }
```

10. 次のコードを挿入して、**C1OrgChart** と **TreeView** 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。

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 = "Main League"
            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#

```
public class League
    {
        public string Name { get; set; }
        public List<Division> Divisions { get; set; }
        public static League GetLeague()
        {
            var league = new League();
            league.Name = "Main League";
            league.Divisions = new List<Division>();
            foreach (var div in "North,South,East,West".Split(','))
            {
                var d = new Division();
                league.Divisions.Add(d);
                d.Name = div;
                d.Teams = new List<Team>();
                foreach (var team in "t1,t2,t3,t4".Split(','))
                {
                    var t = new Team();
                    d.Teams.Add(t);
                    t.Name = string.Format("{0} {1}", team, div);
                }
            }
            return league;
```

11. 次のコードを追加して、チーム、地区、およびリーグ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

```
Public Class Division
        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 Teams() As List(Of Team)
            Get
                Return m_Teams
            End Get
            Set(value As List(Of Team))
                m_Teams = Value
            End Set
        End Property
        Private m_Teams As List(Of Team)
    End Class
    Public Class Team
        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
    End Class
```

C#

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

12. 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。

C1OrgChart: HierarchicalDataTemplate