OutlookBar for WPF/Silverlight

2018.04.11 更新

グレープシティ株式会社

OutlookBar for WPF/Silverlight

目次

製品の概要

このサイドバーナビゲーションシステムを使用して、メニューやコントロールを個別のカテゴリにグループ化できます。**OutlookBar for WPF/Silverlight** は、Microft Outlook のナビゲーションペインに似ています。これには、それぞれがボタンによって表される任意の数のカテゴリが含まれます。カテゴリのボタンには、ヘッダーのほか、コントロールを追加するための1つのパネルがあります。

> **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。

ComponentOne for WPF/Silverlight のヘルプ

はじめに

ComponentOne for WPF/Silverlight のすべてのコンポーネントで共通の使用方法については、「ComponentOne for WPF/Silverlight ユーザーガイド」を参照してください。

主な特長

以下に、**C1OutlookBar** の便利な機能をいくつか示します。

- **Microsoft Outlook 形式の UI**
  Microsoft Outlook 2007/2010 で使用されているナビゲーションシステムのように、コンテンツおよびナビゲーションメニューを個別のカテゴリにグループ化できます。コンテンツが整理されるため、エンドユーザーによるすばやいナビゲーションが可能になります。

- **Office 2007 および 2010 のテーマ**
  C1OutlookBar は、青、黒、シルバーなどのさまざまな Office 2007 および 2010 のテーマをサポートします。これらのテーマは、標準の WPF/Silverlight テーマと同様に機能します。詳細については、このドキュメントの「テーマ」トピックを参照してください。

- **展開/折りたたみ**
  C1OutlookBar コントロールをページの左側または右側に折りたたむことができます。詳細については、「展開と折りたたみ」を参照してください。

- **積み重ね可能なボタン**
  スプリッタバーを使用して、コントロール下部の折りたたみ項目パネルにボタンを積み重ねることができます。ボタンが入りきらない場合は、選択用のドロップダウンメニューに表示されます。

- **柔軟なデータ連結**
  OutlookBar は **ItemsControl** なので、項目のビジュアル表現を作成するために、データテンプレートを使用して任意のデータ型のオブジェクトのリストに連結できます。例については、「C1OutlookItem への画像の追加」トピックを参照してください。

- **ClearStyle を使用して簡単に色を変更する**
  OutlookBar は、テンプレートを上書きしなくてもコントロールのブラシを簡単に変更できる **ComponentOne ClearStyle 技術**をサポートします。Visual Studio でブラシのプロパティをいくつか設定するだけで、アコーディオンの各部のスタイルを簡単に設定できます。ClearStyle 技術の詳細については、「OutlookBar の ClearStyle プロパティ」を参照してください。

クイックスタート

このクイックスタートは、**OutlookBar 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、最初に Visual Studio で新しいプロジェクトを作成し、アプリケーションに **C1OutlookBar** を追加します。次に、メール および タスク という **C1OutlookItem** を追加します。これは、Microsoft Outlook のナビゲーションペインのようになります。

手順 1:アプリケーションの作成

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

プロジェクトをセットアップし、**C1OutlookBar** 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、左ペインから言語を選択し(この例では C# を使用)、テンプレートリストから**[WPF/Silverlight アプリケーション]**を選択します。
3. [.NET Framework]ドロップダウンリストで、**[.NET Framework 4]**以上を選択します。
4. プロジェクトの**名前**を入力し、**[OK]**をクリックします。[新しい WPF/Silverlight アプリケーション]ダイアログボックスが表示されます。
5. 必要に応じて、**[WPF/Silverlight アプリケーションを新しい Web サイトでホストする]**ボックスをオフにし、**[OK]**をクリックします。**MainWindow.xaml** ファイルが開きます。
6. プロジェクトの XAML ウィンドウで、カーソルを <Grid> タグと </Grid> タグの間に置き、1回クリックします。
7. ツールボックスに移動し、**C1OutlookBar** アイコンを <Grid> タグの間にドラッグし、**MainWindow.xaml** に **C1OutlookBar** を追加します。XAML マークアップは次のようになります。

WPF

XAML

```
<Window xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
x:Class="WpfApplication17.MainWindow"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="MainWindow" Height="350" Width="525">
     <Grid>
        <c1:C1OutlookBar />
     </Grid>
</Window>
```

Silverlight

XAML

```
<UserControl x:Class="OutlookBar.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"
```

```
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml" xmlns:my="clr-
namespace:C1.Silverlight;assembly=C1.Silverlight">
    <Grid x:Name="LayoutRoot" Background="White">
        <c1:C1OutlookBar c1:C1NagScreen.Nag="True">
        </c1:C1OutlookBar>
    </Grid>
</UserControl>
```

次の手順では、いくつかの**C1OutlookItem** を**C1OutlookBar** に追加します。

手順 2：C1OutlookItem の追加

続いて、**C1OutlookItem** を**C1OutlookBar** に追加します。ここでは、メール フォルダと タスク フォルダの2つの項目を追加します。

1. XAML マークアップで、<c1:C1OutlookBar></c1:C1OutlookBar> タグの間にカーソルを置き、[Enter]キーを押します。
2. 次の XAML マークアップを使用して、これらのタグ内に**C1OutlookItem** コントロールを追加し、**Header="メール"** を指定します。

   <c1:C1OutlookItem Header="メール"> </c1:C1OutlookItem>

3. 次に、**C1OutlookBar** に表示するフォルダを含む **C1TreeView** を追加します。<c1:C1OutlookItem> タグ内に、次の XAML を追加します。

   XAML

```
    <c1:C1TreeView x:Name="contacts" BorderThickness="0" >
        <c1:C1TreeViewItem Header="受信トレイ" IsExpanded="True">
            <c1:C1TreeViewItem >
                <c1:C1TreeViewItem.Header>
                    <StackPanel Orientation="Horizontal">
                        <TextBlock Text="フォルダ 1"/>
                    </StackPanel>
                </c1:C1TreeViewItem.Header>
            </c1:C1TreeViewItem>
            <c1:C1TreeViewItem >
                <c1:C1TreeViewItem.Header>
                    <StackPanel Orientation="Horizontal">
                        <TextBlock Text="フォルダ 2"/>
                    </StackPanel>
                </c1:C1TreeViewItem.Header>
            </c1:C1TreeViewItem>
        </c1:C1TreeViewItem>
    </c1:C1TreeView>
```

4. **タスク** フォルダを表示するために、**C1TreeView** を含むもう1つの**C1OutlookItem** を追加します。</c1:C1OutlookItem> 終了タグの後に、次の XAML マークアップを加します。

   XAML

```
 <c1:C1TreeView x:Name="tasks" BorderThickness="0" >
```

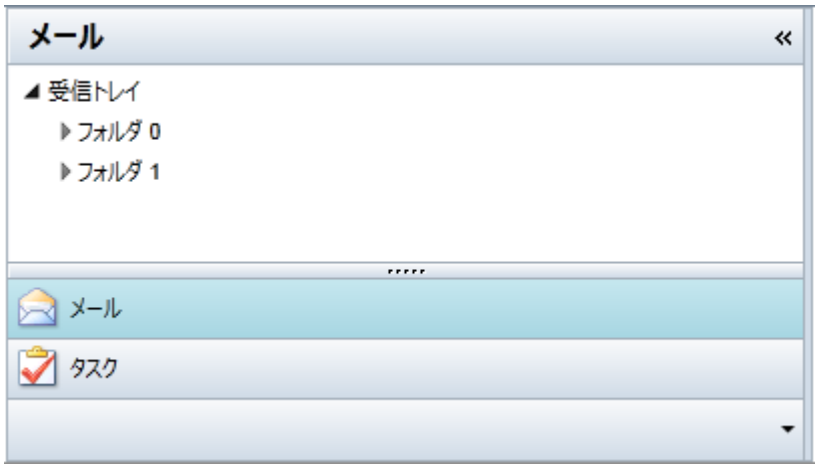
```
            <c1:C1TreeViewItem Header="個人用のタスク" IsExpanded="True" >
                <c1:C1TreeViewItem >
                    <c1:C1TreeViewItem.Header>
                        <StackPanel Orientation="Horizontal">
                            <TextBlock Text="To-Do バーのタスクリスト"/>
                        </StackPanel>
                    </c1:C1TreeViewItem.Header>
                </c1:C1TreeViewItem>
                <c1:C1TreeViewItem >
                    <c1:C1TreeViewItem.Header>
                        <StackPanel Orientation="Horizontal">
                            <TextBlock Text="タスク"/>
                        </StackPanel>
                    </c1:C1TreeViewItem.Header>
                </c1:C1TreeViewItem>
            </c1:C1TreeViewItem>
        </c1:C1TreeView>
    </c1:C1OutlookItem>
```

次の手順では、このアプリケーションを実行します。

手順 3:アプリケーションの実行

C1OutlookBar を含むアプリケーションを作成したので、次に、このアプリケーションを実行します。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ように表示されます。



2. タスク **C1OutlookItem** をクリックして、この項目に切り替えます。

おめでとうございます!

これで、**OutlookBar for WPF** クイックスタートは終了です。

XAML クイックリファレンス

このトピックでは、**C1OutlookBar** および**C1OutlookItem** の作成に使用される XAML の概要を提供します。

開発を開始するには、ルート要素タグに **c1** 名前空間宣言を追加します。

xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"

次の例は、複数の**C1OutlookBar** 項目を含む基本的な**C1OutlookBar** です。

XAML

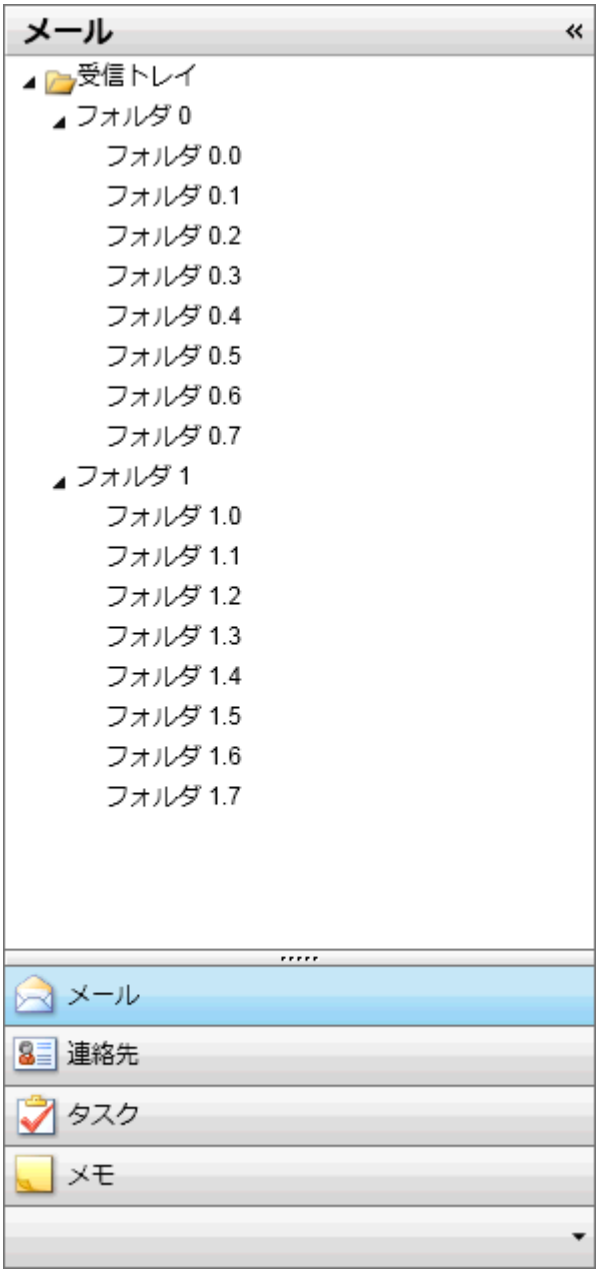
```
<c1:C1OutlookBar>
  <c1:C1OutlookItem />
  <c1:C1OutlookItem />
  <c1:C1OutlookItem />
</c1:C1OutlookBar>
```

次の XAML は、これより複雑な**C1OutlookBar** の例です。複数の**C1OutlookItem** にアイコンが含まれています。

XAML

```
<c1:C1OutlookBar IsExpanded="True" FontFamily="Arial" MinWidth="36"
ExpandedWidth="300" c1:C1NagScreen.Nag="True">
   <!-- メール -->
     <c1:C1OutlookItem Header="メール" LargeIcon="Resources/Inbox24.png"
SmallIcon="Resources/Inbox.png" c1:C1NagScreen.Nag="True">
         <userControls:Inbox >
     </c1:C1OutlookItem>
     <!-- 連絡先 -->
     <c1:C1OutlookItem Header="連絡先" LargeIcon="Resources/Contacts24.png"
SmallIcon="Resources/Contacts.png" c1:C1NagScreen.Nag="True">
         <userControls:Contacts >
     </c1:C1OutlookItem>
     <!-- タスク -->
     <c1:C1OutlookItem Header="タスク" LargeIcon="Resources/Tasks24.png"
SmallIcon="Resources/Tasks.png" c1:C1NagScreen.Nag="True">
         <userControls:Tasks >
     </c1:C1OutlookItem>
     <!-- メモ -->
     <c1:C1OutlookItem Header="メモ" LargeIcon="Resources/Notes24.png"
SmallIcon="Resources/Notes.png" c1:C1NagScreen.Nag="True">
         <userControls:Notes >
     </c1:C1OutlookItem>
</c1:C1OutlookBar>
```

この XAML の結果は次のようになります。



C1OutlookBarの操作

OutlookBar for WPF/Silverlight には、Microsoft Outlook のようなサイドバーナビゲーションシステムを提供する**C1OutlookBar** コントロールが含まれます。以下のトピックでは、**C1OutlookBar** コントロールと、このコントロールに追加できる要素について詳しく説明します。

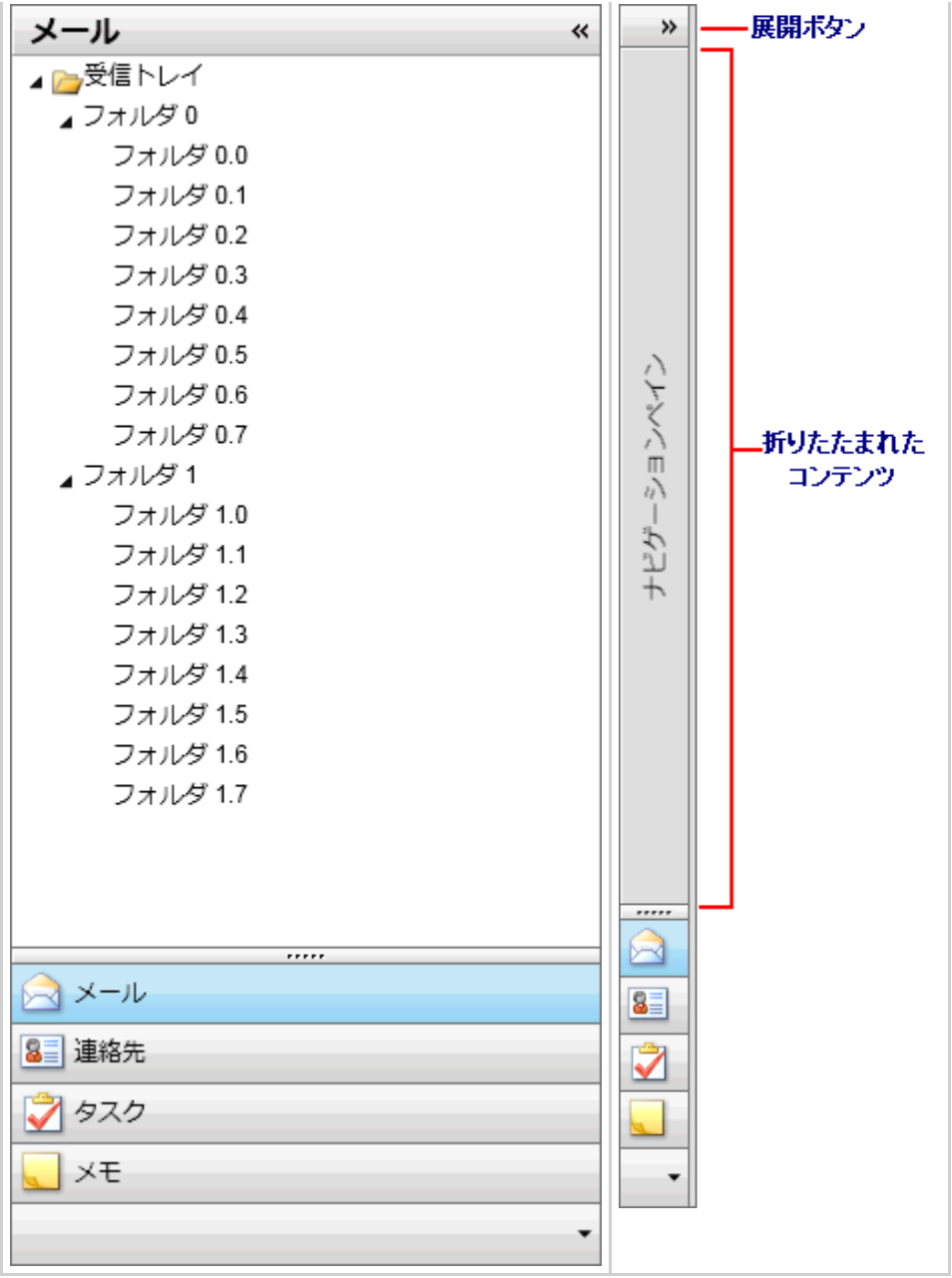
OutlookBar の要素

C1OutlookBar は、アイコン付きの**C1OutlookItem** を組み込んでシンプルなナビゲーションシステムを作成するためのコンテナとして機能します。**C1OutlookBar** は次の要素で構成されます。

C1OutlookBar の展開と折りたたみ

OutlookBar for WPF/Silverlight は、他のコントロールや要素のための領域を確保できるように、**C1OutlookBar** を展開/折りたたむ機能を提供します。次に、展開された状態と折りたたまれた状態の**C1OutlookBar** を示します。

展開された状態	折りたたまれた状態

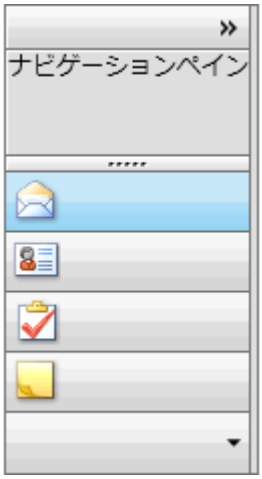


折りたたみコンテンツ

C1OutlookBar を折りたたむと、中心となるコンテンツ領域に、カスタマイズされた折りたたみコンテンツを表示できます。Microsoft Outlook 2007 ～ 2010 の場合、この領域には、**「ナビゲーションペイン」**というテキストが縦方向に表示されます。この領域を選択した項目に関連付けて、柔軟性を高めることができますが、必須ではありません。折りたたみコンテンツを表示するには、次の2つのオプションがあります。

オプション A：CollapsedContent プロパティを使用して、この**C1OutlookBar** が折りたたまれたときのテキストを表示します。

```
<c1:C1OutlookBar CollapsedContent = "ナビゲーションペイン" />
```



オプション B：CollapsedContent プロパティを使用して、選択された項目とは関係なく、同じ折りたたみコンテンツを表示します。

XAML

```
<c1:C1OutlookBar>
   <c1:C1OutlookBar.CollapsedContent>
   </c1:C1OutlookBar.CollapsedContent>
</c1:C1OutlookBar>
```

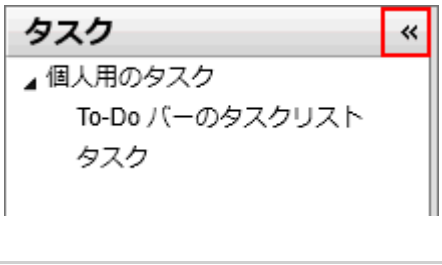
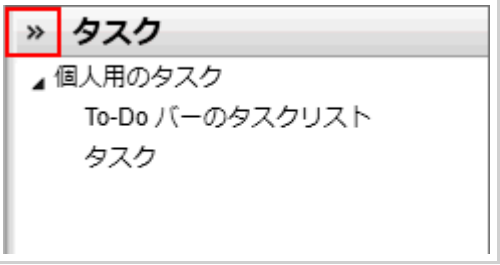
たとえば、次の XAML は、**「ナビゲーションペイン」**というテキストを縦方向に表示します。

XAML

```
<c1:C1OutlookBar.CollapsedContent>
             <c1:C1LayoutTransformer>
                <c1:C1LayoutTransformer.LayoutTransform>
                   <RotateTransform Angle="270" />
                </c1:C1LayoutTransformer.LayoutTransform>
                <TextBlock FontSize="13" TextAlignment="Center"
VerticalAlignment="Center"
                        Text="ナビゲーションペイン" />
            </c1:C1LayoutTransformer>
          </c1:C1OutlookBar.CollapsedContent>
```

折りたたみ方向

C1OutlookBar は、左方向（デフォルト）と右方向の折りたたみをサポートします。**C1OutlookBar** のドッキングが Left（デフォルト）に設定された場合は、コントロールの右側に折りたたみ矢印が表示されます。**C1OutlookBar** のドッキングが Right に設定された場合は、コントロールの左側に折りたたみ矢印が表示されます。

Dock = Left	Dock = Right
	

C1OutlookBar の折りたたみ方向を指定するには、**Dock** プロパティを使用します。

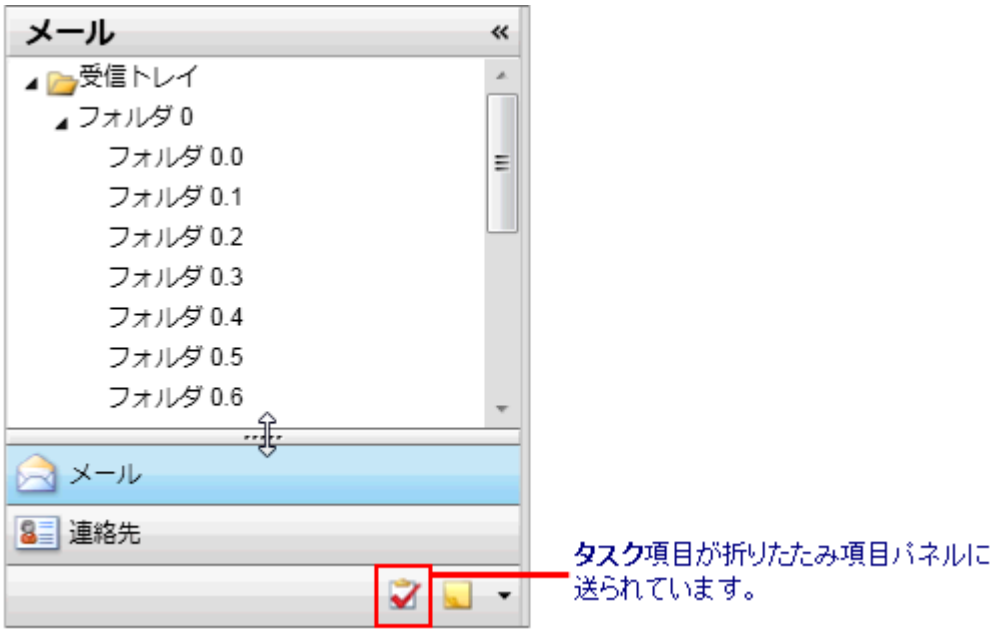
```
<c1:C1OutlookBar x:Name="C1OutlookBar1" Dock="Right">
```

また、Visual Studio の［プロパティ］ウィンドウで、**Dock** プロパティを設定することもできます。

C1OutlookBar の項目の折りたたみ

項目の折りたたみと**C1OutlookBar** コントロールの折りたたみとは違います。**C1OutlookBar** の**項目コンテンツ**領域と**項目ボタン**領域の間には、ドラッグ可能なスプリッタバーがあります。このスプリッタバーを下にドラッグすると、いくつかの項目が折りたたみ項目パネルに折りたたまれます。

次の画像では、スプリッタバーが下にドラッグされて、**タスク** 項目が折りたたみ項目パネルに送られています。



C1OutlookBar は、いくつかの**項目を項目ボタン**に表示し、いくつかの**項目を折りたたみ項目パネル**に折りたたんで表示することができます。**FirstOverflowIndex** プロパティを使用すると、**折りたたみ項目パネル**に送る項目をインデックスで指定できます。

たとえば、上の画像では、**タスク** 項目は**C1OutlookBar** の3番目の項目でした。**FirstOverflowIndex** プロパティを2に設定すると、最初の2つ以外の項目（オーバーフロー項目）が**折りたたみ項目パネル**に送られます。

FirstOverflowIndex プロパティは、XAML で設定できます。

XAML

```
<c1:C1OutlookBar FirstOverflowIndex="2">
```

または、Visual Studio の[プロパティ]ウィンドウで設定することもできます。

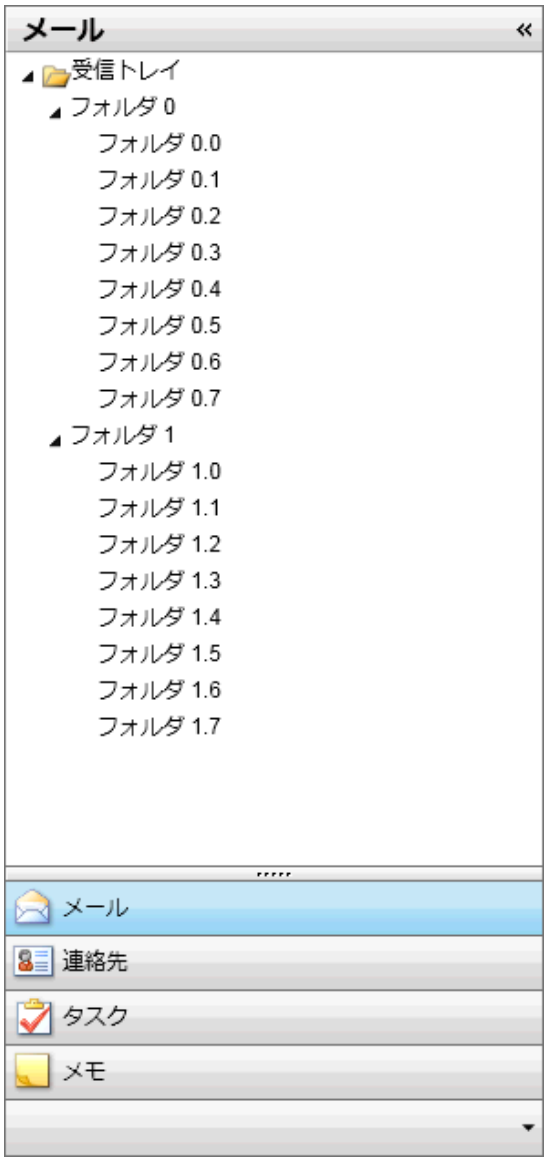
レイアウトおよび外観

以下のトピックでは、**C1OutlookBar** 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。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、コントロールの動作をカスタマイズできます。テーマを使用することで、コントロールの外観をカスタマイズしたり、WPF/Silverlight の XAML ベースのスタイル設定を活用することができます。

テーマ

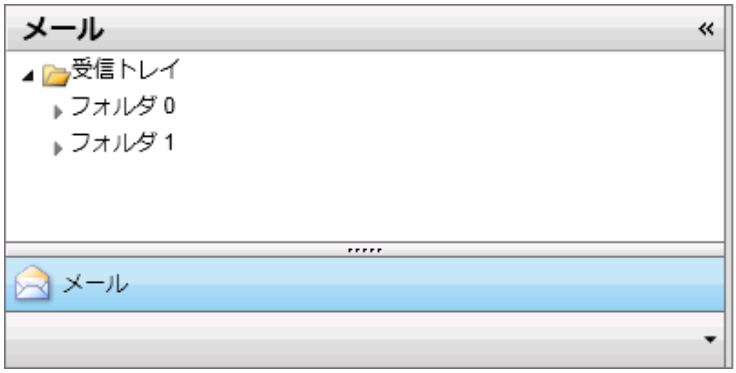
WPF/Silverlight のテーマは、いくつかのコントロールの外観を定義するイメージ設定のコレクションです。テーマはアプリケーション内の複数のコントロールに適用できるため、テーマを使用すると、スタイル設定作業を繰り返さなくても、一貫性のあるコントロールを作成できます。

C1OutlookBar をプロジェクトに追加すると、C1OutlookBar は次の図のように、デフォルトのテーマで表示されます。

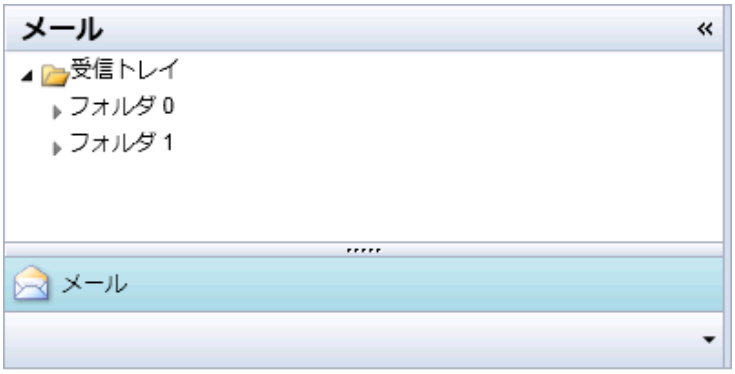
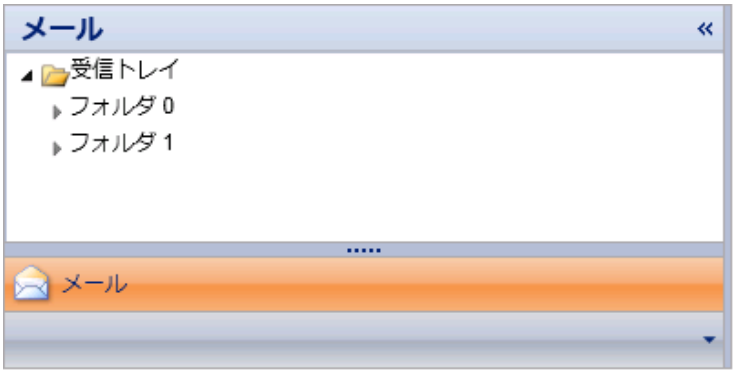


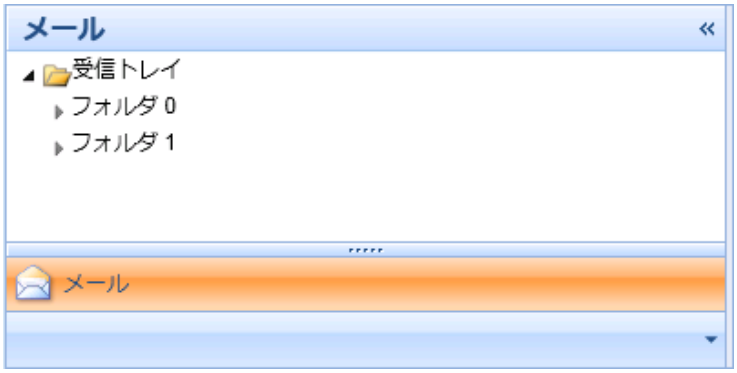
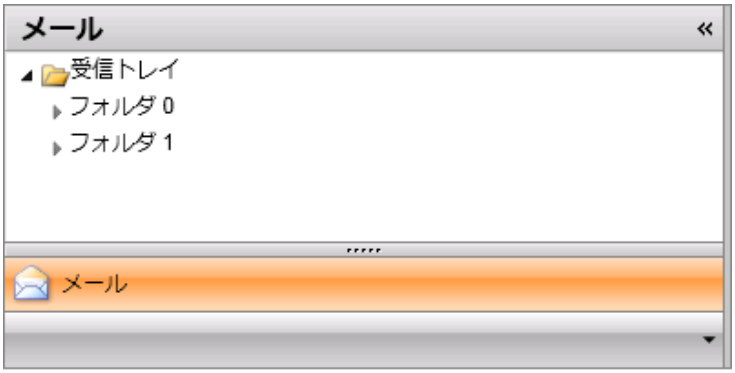
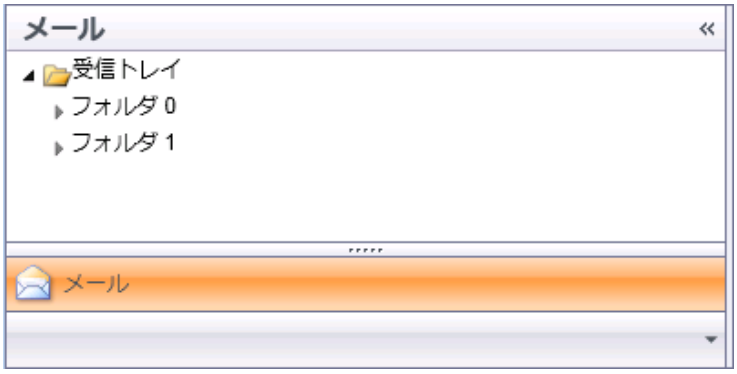
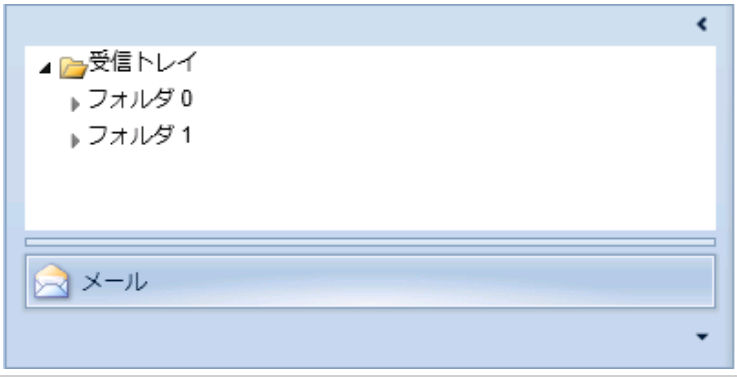
C1OutlookBar は、Office 2010 および Office 2007 のテーマをサポートします。付属する WPF/Silverlight のテーマの1つを設定できます。これらのテーマには、Office2010（C1ThemeOffice2010Blue、C1ThemeOffice2010Black、および C1ThemeOffice2010Silver）、Office2007（C1ThemeOffice2007Blue、C1ThemeOffice2007Black、および C1ThemeOffice2007Silver）、C1ThemeBureauBlack、C1ThemeExpressionDark、C1ThemeExpressionLight、C1ThemeCosmopolitan、C1ThemeRanierOrange、C1ThemeShinyBlue、および C1ThemeWhistlerBlue があります。C1OutlookBar のテーマ

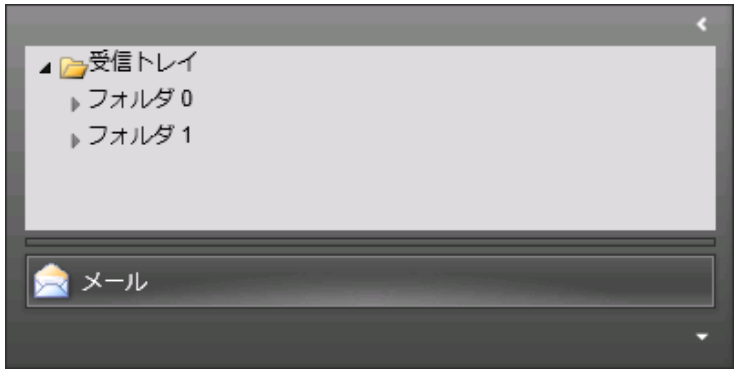
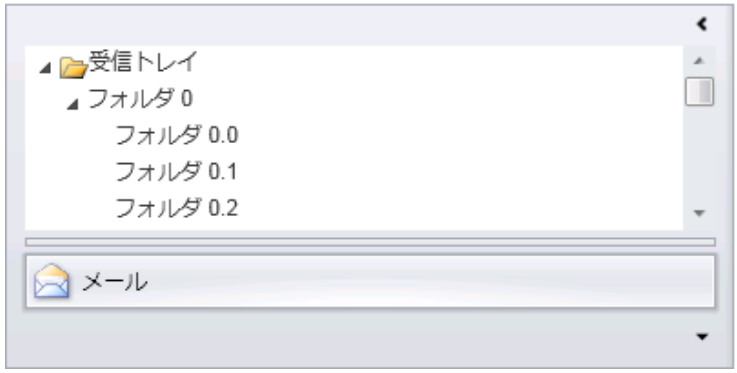
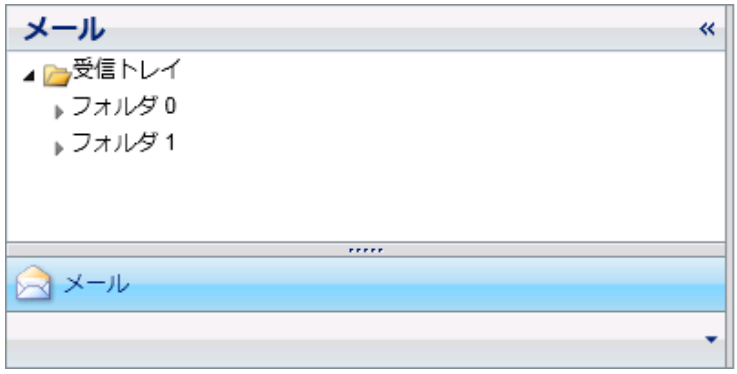
Outlookbar for WPF/Silverlight には、グリッドの外観をカスタマイズできるいくつかのテーマが組み込まれています。ページに **C1OutlookBar** および **C1OutlookItem** コントロールを追加すると、次の画像のようになります。



これは、このコントロールのデフォルトの外観です。この外観は、組み込みテーマの1つを使用したり、独自のカスタムテーマを作成することで変更できます。すべての組み込みテーマは、WPF/Silverlight Toolkit テーマに基づいています。以下に、組み込みのテーマについて説明および図示します。

テーマ名	テーマのプレビュー
C1Blue (WPF のみ)	
C1ThemeBureauBlack	
C1ThemeExpressionDark	

C1ThemeExpressionLight	
C1ThemeOffice2007Blue	
C1ThemeOffice2007Black	
C1ThemeOffice2007Silver	
C1ThemeOffice2010Blue	

C1ThemeOffice2010Black	
C1ThemeOffice2010Silver	
C1ThemeShinyBlue	
C1ThemeWhistlerBlue	

要素のテーマを設定するには、**ApplyTheme** メソッドを使用します。最初に、テーマアセンブリへの参照をプロジェクトに追加し、次のようにコードでテーマを設定します。

Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
    Dim theme As New C1ThemeExpressionDark
    ' ApplyTheme の使用
    C1Theme.ApplyTheme(LayoutRoot, theme)
```

C#

```
private void Window_Loaded(object sender, RoutedEventArgs e)
  {
    C1ThemeExpressionDark theme = new C1ThemeExpressionDark();
    //ApplyTheme の使用
    C1Theme.ApplyTheme(LayoutRoot, theme);
  }
```

アプリケーション全体にテーマを適用するには、**System.Windows.ResourceDictionary.MergedDictionaries** プロパティを使用します。最初に、テーマアセンブリへの参照をプロジェクトに追加し、次のようにコードでテーマを設定します。

Visual Basic

```
Private Sub Window_Loaded(sender As System.Object, e As System.Windows.RoutedEventArgs)
Handles MyBase.Loaded
    Dim theme As New C1ThemeExpressionDark
    ' MergedDictionaries の使用
Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme))
End Sub
```

C#

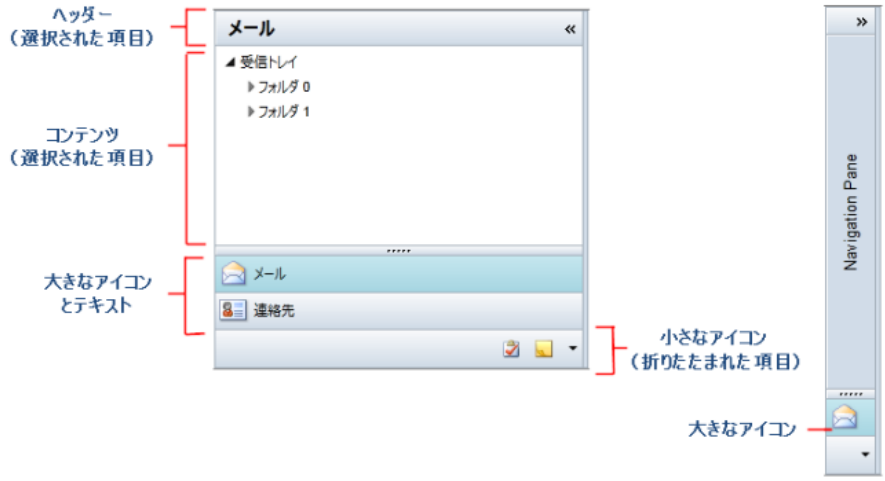
```
private void Window_Loaded(object sender, RoutedEventArgs e)
  {
     C1ThemeExpressionDark theme = new C1ThemeExpressionDark();
     //MergedDictionaries の使用

Application.Current.Resources.MergedDictionaries.Add(C1Theme.GetCurrentThemeResources(theme));
  }
```

この方法は、初めてテーマを適用する場合にのみ使用できることに注意してください。別の ComponentOne テーマに切り替える場合は、最初に、**Application.Current.Resources.MergedDictionaries** から前のテーマを削除します。

テンプレート

C1OutlookBar および**C1OutlookItem** のいくつかの基本的なプロパティを設定して、各**C1OutlookItem** の外観をカスタマイズできます。これらのプロパティにはそれぞれ、そのプロパティを連結するための **Template** プロパティがあります。

プロパティ	対応するテンプレートプロパティ	説明
Header	HeaderTemplate	バーを展開すると、コンテンツの上の **LargeIcon** の右に、Office 2007 スタイルで **Header** が表示されます。
Content	ContentTemplate	この項目が選択されたときに**項目コンテンツ**セクションに表示するコンテンツを取得または設定します。
LargeIcon	LargeIconTemplate	**LargeIcon** は、**C1OutlookBar** が折りたたまれたときに垂直スタックに表示される **C1OutlookItem** の画像です。
SmallIcon	SmallIconTemplate	**FirstOverflowIndex** より大きな項目または**項目ボタン**リストに入りきらない項目は、**SmallIcon** を使用して、**LargeIcon** の下の水平ラインに配置されます。



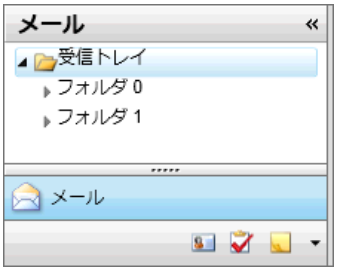
メモ: **SmallIcon** および **Header** を使用するメニューでは、個々の項目を表示または非表示にできます。これらのプロパティは複数の場所に表示されるため、これには `UIElement` を割り当てないでください。

次は、Header が「メール」と設定された C1OutlookItem を含む C1OutlookBar の例を示します。小さなアイコンと大きなアイコンは、対応するテンプレートを使用して連結されます。

XAML

```
<c1:C1OutlookBar.LargeIconTemplate>
    <DataTemplate>
        <Image Source="{Binding}" />
    </DataTemplate>
</c1:C1OutlookBar.LargeIconTemplate>
<c1:C1OutlookBar.SmallIconTemplate>
    <DataTemplate>
        <Grid Height="24">
             <Image Source="{Binding}" />
        </Grid>
    </DataTemplate>
</c1:C1OutlookBar.SmallIconTemplate>
<c1:C1OutlookItem Header="メール" LargeIcon="Resources/Inbox24.png" SmallIcon="Resources/Inbox.png">
    <userControls:Inbox />
</c1:C1OutlookItem>
```

C1OutlookBar は次のようになります。



ComponentOne ClearStyle 技術

ComponentOne ClearStyle は、WPF/Silverlight コントロールのスタイル設定をすばやく簡単に実行できる新技術です。ClearStyle を使用すると、面倒な XAML テンプレートやスタイルリソースを操作しなくても、コントロールのカスタムスタイルを作成できます。

現在のところ、すべての標準 WPF/Silverlight コントロールにテーマを追加するには、スタイルリソーステンプレートを作成する必要があります。Microsoft Visual Studio ではこの処理は困難であるため、Microsoft は、このタスクを簡単に実行できるように Expression Blend を導入しました。ただし、Blend に不慣れであったり、十分な学習時間を取れない開発者にとっては、この2つの環境を行き来することはかなり困難な作業です。デザイナに作業を任せることも考えられますが、デザイナと開発者

が XAML ファイルを共有すると、かえって煩雑になる可能性があります。

このような場合に、ClearStyle を使用します。ClearStyle は、Visual Studio を使用して直感的な方法でスタイル設定を実行できるようにします。ほとんどの場合は、アプリケーション内のコントロールに対して単純なスタイル変更を行うだけなので、この処理は簡単に行えるべきです。たとえば、データグリッドの行の色を変更するだけであれば、1つのプロパティを設定するだけで簡単に行えるようにする必要があります。一部の色を変更するためだけに、完全で複雑なテンプレートを作成する必要はありません。

ClearStyle の仕組み

コントロールのスタイルの主な要素は、それぞれ単純な色プロパティとして表されます。これが集まって、コントロール固有のスタイルプロパティセットを形成します。たとえば、**Gauge** には **PointerFill** プロパティや **PointerStroke** プロパティがあり、**DataGrid** の行には **SelectedBrush** や **MouseOverBrush** があります。

たとえば、フォーム上に ClearStyle をサポートしていないコントロールがあるとします。その場合は、ClearStyle によって作成された XAML リソースを使用して、フォーム上の他のコントロールを調整して合わせることができます（正確な色合わせなど）。また、スタイルセットの一部を ClearStyle（カスタムスクロールバーなど）で上書きしたいとします。ClearStyle は拡張可能なのでこれも可能です。必要な場所でスタイルを上書きできます。

ClearStyle は、すばやく簡単にスタイルを変更することを意図したソリューションですが、ComponentOne のコントロールには引き続き従来の方法を使用して、必要なスタイルを細かく指定して作成できます。完全なカスタム設計が必要になる特別な状況で ClearStyle が邪魔になることはありません。

OutlookBar の ClearStyle プロパティ

OutlookBar for WPF/Silverlight は、コントロールのテンプレートを変更することなくコントロールの色を簡単に変更できる ComponentOne の新しい ClearStyle 技術をサポートします。色のプロパティをいくつか設定するだけで、**C1OutlookBar** 要素のスタイルを簡単に設定できます。次の表に、**C1OutlookBar** でサポートされているプロパティを一覧します。

プロパティ	説明
Background	**C1OutlookBar** の塗りつぶしに使用される背景を取得または設定します。
Foreground	**C1OutlookBar** の塗りつぶしに使用される前景を取得または設定します。
MouseOverBrush	マウスポインタが置かれた項目を強調表示するために使用されるブラシを取得または設定します。
SelectedBackground	**C1OutlookBar** で選択された**C1OutlookItem** の塗りつぶしに使用されるブラシを取得または設定します。

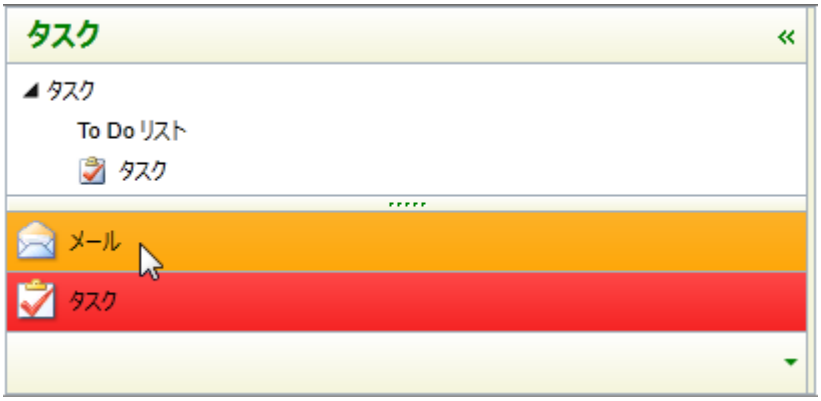
これらのプロパティを設定することで、**C1OutlookBar** の外観を完全に変更できます。たとえば、**ItemBackground** プロパティを **Beige**、**ItemForeground** を **Green**、**MouseOverBrush** を **Orange**、**SelectedBackground** を **Red** に設定する場合、XAML マークアップは次のようになります。

XAML

```
<c1:C1OutlookBar Background="Beige" Foreground="Green"
MouseOverBrush="Orange" SelectedBackground="Red" x:Name="C1OutlookBar1">
```

C1OutlookBar は次のように表示されます。

OutlookBar for WPF/Silverlight



タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio でのプログラミングに精通しており、**C1OutlookBar** 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。**OutlookBar 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、**OutlookBar 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。

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

C1OutlookItem への画像の追加

C1OutlookBar は **ItemsControl** なので、項目のビジュアル表現を作成するために、データテンプレートを使用して任意のデータ型のオブジェクトのリストに連結できます。この例では、**DataTemplate** を使用した連結によって**C1OutlookItem** に簡単に画像を追加する方法について説明します。

Header = メール とした**C1OutlookItem** を含む **C1OutlookBar1** という名前の**C1OutlookBar** があるとします。この**C1OutlookItem** の画像も必要です。この例では、Resources フォルダにプロジェクトの画像が含まれるとします。XAML は次のようになります。

XAML

```
<c1:C1OutlookBar Height="274" HorizontalAlignment="Left" Margin="56,12,0,0"
Name="c1OutlookBar1"
VerticalAlignment="Top" Width="274">
       <c1:C1OutlookItem Header="メール"/>
</c1:C1OutlookBar>
```

1. 折りたたみ**項目パネル**および**項目ボタン**に表示される画像にそれぞれ対応する**SmallIconTemplate** および**LargeIconTemplate** の XAML を追加します。このマークアップは、**C1OutlookBar** タグ内に配置します。

   XAML

   ```
   <c1:C1OutlookBar.LargeIconTemplate>
         <DataTemplate>
               <Image Source="{Binding}" Width="24" Height="24" />
         </DataTemplate>
   </c1:C1OutlookBar.LargeIconTemplate>
   <c1:C1OutlookBar.SmallIconTemplate>
         <DataTemplate>
               <Grid Height="24">
                     <Image Source="{Binding}" Width="16" Height="16" />
               </Grid>
         </DataTemplate>
   </c1:C1OutlookBar.SmallIconTemplate>
   ```

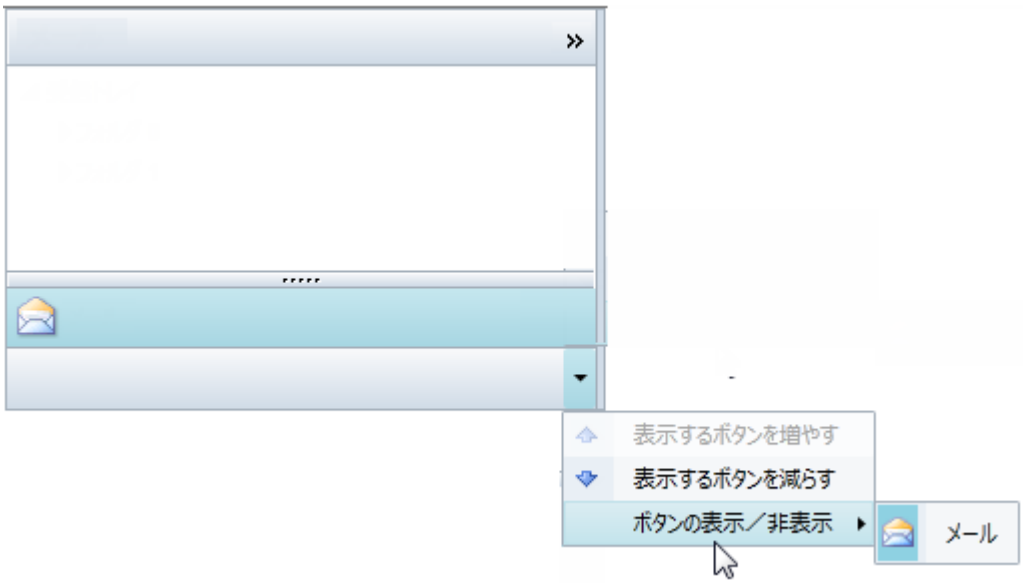
2. データテンプレートに連結する**SmallIcon** および**LargeIcon** を指定する XAML を**C1OutlookItem** に追加します。マークアップは次のようになります。

   <c1:C1OutlookItem LargeIcon="Resources/Inbox24.png" SmallIcon="Resources/Inbox.png" Header="メール"/>

3. 3.プロジェクトを実行します。**C1OutlookBar** を折りたたむと、折りたたみ**項目パネル**に小さなアイコンが表示されます。**C1OutlookBar** を展開すると、**項目ボタン(展開)**領域に大きなアイコンが表示されます。

展開された状態

折りたたまれた状態



C1OutlookBar の項目の選択

C1OutlookBar は、単一項目の選択機能をサポートし、プログラムで項目を選択できます。ユーザーが初めて**C1OutlookBar** を表示したときに、どの項目が選択されるかを制御できます。

いくつかの**C1OutlookItem** を含む **C1OutlookBar1** という名前の**C1OutlookBar** があるとします。項目の1つには selectedItem という名前を付けます。

XAML

```
<c1:C1OutlookBar x:Name="C1OutlookBar1 ">
<c1:C1OutlookItem x:Name="selectedItem" Header="タスク">
```

1. Visual Stuido のメニューから**[表示]**→**[コード]**を選択します。
2. プロジェクトに次のコードを追加します。

XAML

```
using C1.WPF.OutlookBar;
namespace C1OutlookBar
{
  public partial class MainPage : UserControl
```

```
    {
      public MainPage()
      {
        InitializeComponent();
        C1OutlookBar1.SelectedItem = (C1OutlookItem)selectedItem;
      }
    }
}
```

3. プロジェクトを実行します。**selectedItem** という名前の**C1OutlookItem**（この例では、**タスク** 項目）が選択されます。