DockControl　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

DockControl for WPF/Sliverlight を使用すると、WPF アプリケーションで複数のウィンドウを操作できます。Microsoft Visual Studio のドッキングシステムと同様に、**C1DockControl** は、ドッキング可能なウィンドウ、フローティングウィンドウ、およびタブ付きウィンドウを提供します。また、セクションを自動的に非表示にしたり、**C1DockControl** のスタイルを簡単に設定できます。

DockControl for WPF/Silverlightに含まれるコントロール:

- **C1DockControl**
- **C1DockTabControl**

> **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、**C1DockControl** の便利な機能をいくつか示します。

- **ひし形のドッキングガイド**
  デフォルトでは、**DockControl** は Microsoft の Visual Studio と同様のひし形のドッキングガイドを使用します。詳細については、「ひし形のドッキングガイドとゾーン」を参照してください。

- **わかりやすいドッキング機能**
  DockControl をカスタマイズして、アプリケーション内で複数のウィンドウをドッキングしたり浮動化できる、わかりやすいドッキング機能を提供できます。ドッキングゾーン上でウィンドウをドラッグすると、ドッキングインジケータが表示され、ドッキング先を示します。

- **フローティングウィンドウ**
  アプリケーション内で、ウィンドウを独立したウィンドウとして浮動化できます。

- **タブ付きウィンドウ**
  エディタのインスタンス内で開かれるドキュメントは、自動的にタブ付きペインに配置されます。

- **自動非表示**
  押しピンによる自動非表示機能を使用すると、一度により多くのコードを表示できます。この機能により、使用していないツールウィンドウを IDE の端に最小化できます。

- **豊富な機能を持つプログラム API**
  高い柔軟性を保ちつつ動作を制限するために、ドッキングウィンドウのドッキング状態は、豊富な機能を持つプログラム API を使用してプログラムから制御できます。

- **2つのモニタのサポート(WPF のみ)**
  1つのモニタから別のモニタにウィンドウをドラッグするだけで、ウィンドウを表示するモニタを選択できます。このオプションは、WPF プラットフォームでのみ使用できます。

- **ClearStyle 技術のサポート**
  DockControl 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、**C1DockControl** 要素のスタイルを簡単に設定できます。ComponentOne ClearStyle 技術の詳細については、「DockControl 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、**DockControl 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、**C1DockControl** をアプリケーションに追加してから、**C1DockTabItem** を含む**C1DockTabControl** を追加します。

手順 1:アプリケーションの作成

この手順では、Visual Studio で、**DockControl for WPF/Silverlight** を使用して、WPF/Silverlight 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、**C1DockControl** 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/Sliverlight 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 3.5]**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/Sliverlight 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/Sliverlight 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。**MainPage.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、**[C1DockControl]**アイコンを <Grid> タグの間にドラッグし、**Window.xaml** に **C1DockControl** 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="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:c1="http://schemas.componentone.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
  <Grid x:Name="LayoutRoot">
<c1:C1DockControl></c1:C1DockControl>
  </Grid>
</Window>
```

Silverlight

XAML

```
<UserControl xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
x:Class="C1DockControlQuickStart.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:DesignWidth="640" d:DesignHeight="480">
    <Grid x:Name="LayoutRoot">
        <c1:C1DockControl></c1:C1DockControl>
    </Grid>
</UserControl>
```

メモ:C1.WPF.Docking または C1.Silverlight.Docking 名前空間と<c1:C1DockControl></c1:C1DockControl> タグがプロジェクトに追加されています。

次の手順では、**C1DockTabItem** を含む **C1DockTabControl** を追加します。

手順 2:C1DockTabItems を含む C1DockTabControl の追加

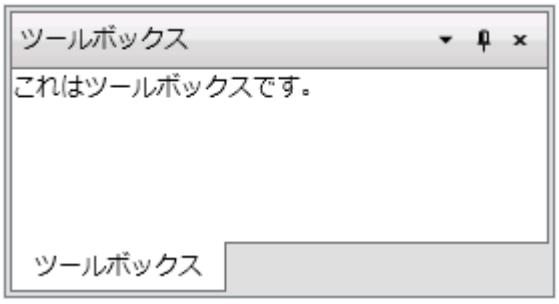
ここでは、**C1DockTabItem** を含む **C1DockTabControl** を追加します。

1. XAML マークアップで、<c1:C1DockControl></c1:C1DockControl> 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 **C1DockTabControl** を追加します。

XAML

```
<c1:C1DockTabControl Dock="Left">
    <c1:C1DockTabItem Header="ツールボックス">
        <TextBlock Text="これはツールボックスです。" />
    </c1:C1DockTabItem>
</c1:C1DockTabControl>
```

この XAML には、**Toolbox** のラベルが付けられた **C1DockTabItem** も含まれています。ここでアプリケーションを実行すると、**C1DockTabControl** は次のように表示されます。



アプリケーションを実行したときに、タブを選択したり、**C1DockTabControl** をドッキングまたは浮動化できることを確認できるように、**C1DockTabItem** と **C1DockTabControl** をもう1つ追加します。

3. </c1:C1DockTabItem> の終了タグの後に、**C1DockTabItem** と **C1DockTabControl** の XAML をもう1つ追加します。**C1DockControl** の XAML 全体は次のようになります。

XAML

```
<c1:C1DockControl>
    <c1:C1DockTabControl Dock="Left">
        <c1:C1DockTabItem Header="ツールボックス">
```

```
            <TextBlock Text="これはツールボックスです。" />
        </c1:C1DockTabItem>
        <c1:C1DockTabItem Header="プロジェクト">
            <TextBlock Text="ファイルツリー" />
        </c1:C1DockTabItem>
    </c1:C1DockTabControl>
    <c1:C1DockTabControl DockWidth="500" ShowHeader="False">
        <c1:C1DockTabItem Header="myfile.ext">
            <TextBlock Text="これは myfile.ext のテキストです。" />
        </c1:C1DockTabItem>
    </c1:C1DockTabControl>
</c1:C1DockControl>
```

次の手順では、アプリケーションを実行して、ウィンドウのドッキングモードを変更してみます。

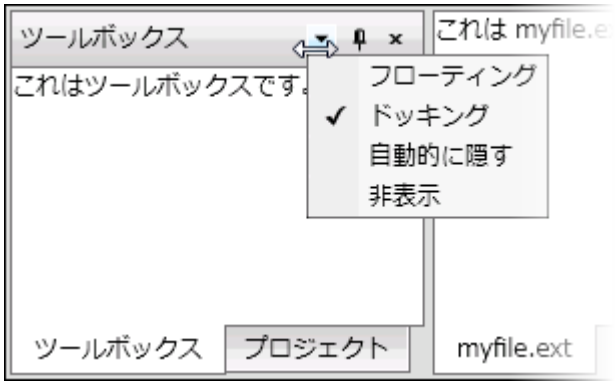
手順 3：アプリケーションの実行

C1DockControl とタブ項目を含む WPF/Silverlight 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。

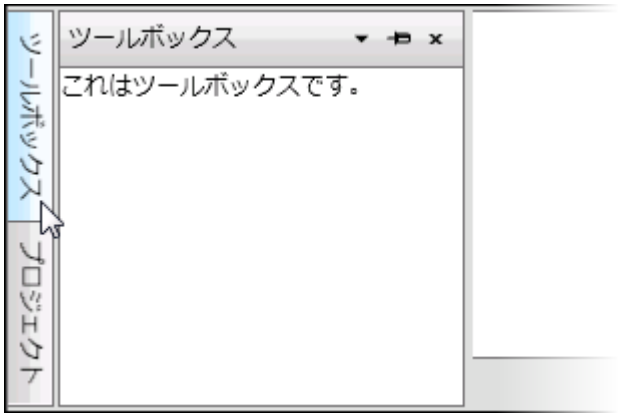
1. **[デバッグ]**メニューから**[デバッグ開始]**を選択し、実行時にアプリケーションがどのように表示されるかを確認します。アプリケーションは次のように表示されます。

[ツールボックス]は、XAML で指定したように、左側にドッキングされています。

2. **[ツールボックス]**のヘッダーにあるドロップダウン矢印をクリックし、オプションを選択して、**C1DockControl** を浮動化、スライド、または非表示にしてみます。

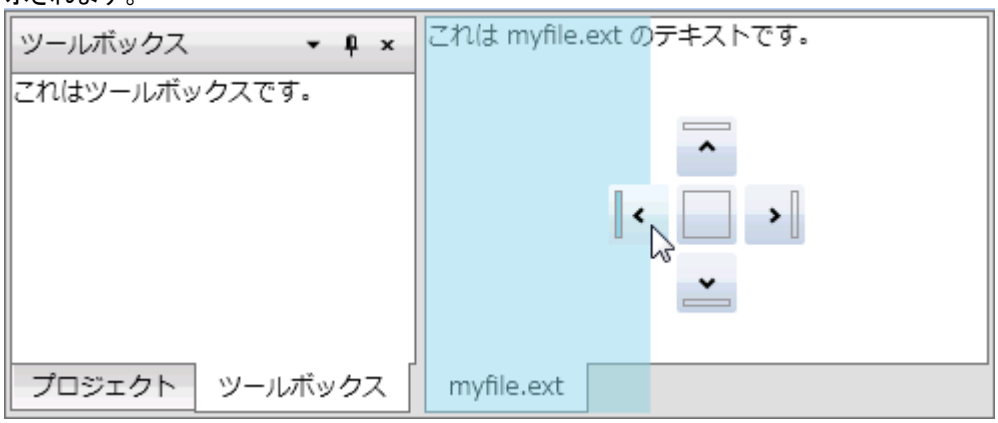


C1DockControl のこれらのオプションは、**[プロジェクト]**タブをクリックして選択することもできます。押しピン をクリックすると **C1DockTabItems** が左側に表示され、タブをクリックするとスライドして開きます。



[非表示]ボタン をクリックすると、**C1DockControl** は非表示になります。

3. **[ツールボックス]**のヘッダーを選択し、[myfile.ext]タブがある **C1DockControl** 上にドラッグします。ドッキングインジケータがドッキングゾーン上に表示されます。いずれかのインジケータ上にポインタを置くと、ドッキングゾーンが青で表示されます。



ShowHeader を **False** に設定したので、myfile.ext **C1DockTabItem** がある2番目の **C1DockControl** にはヘッダーがありません。

おめでとうございます!

これで、**DockControl for WPF/Sliverlight** クイックスタートは終了です。このクイックスタートでは、**C1DockControl** を使用して、浮動化やドッキングが可能なウィンドウを作成しました。

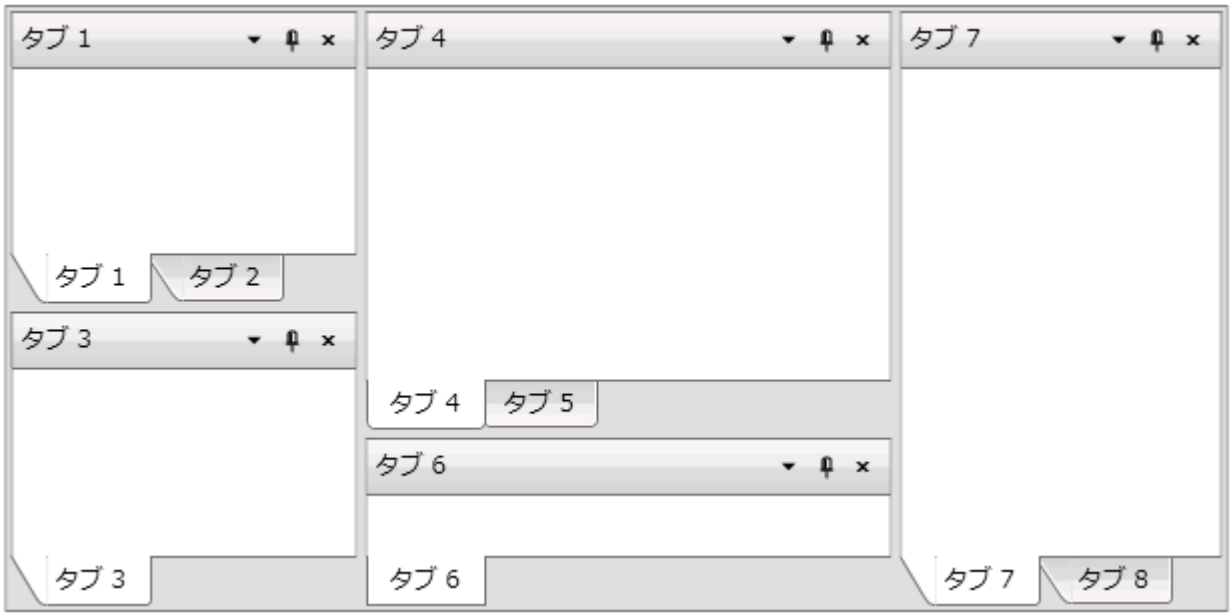
XAML クイックリファレンス

このトピックでは、**C1DockControl**、**C1DockGroup**、**C1DockTabControl**、および **C1DockTabItem** を作成するための 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"

次は、サンプルドッキングレイアウトです。



このサンプルドッキングレイアウトの XAML は次のようになります。

XAML

```
                <c1:C1DockControl x:Name="dockControl" Margin="-128,0,12,127">
            <c1:C1DockGroup>
                <c1:C1DockTabControl Dock="Top">
                    <c1:C1DockTabItem Header="タブ 1" TabShape="Sloped">
                        <!--ここにコンテンツを配置-->
                    </c1:C1DockTabItem>
                    <c1:C1DockTabItem Header="タブ 2" TabShape="Sloped">
                        <!--ここにコンテンツを配置-->
                    </c1:C1DockTabItem>
                </c1:C1DockTabControl>
                <c1:C1DockTabControl>
                    <c1:C1DockTabItem Header="タブ 3" TabShape="Sloped" >
                        <!--ここにコンテンツを配置-->
                    </c1:C1DockTabItem>
                </c1:C1DockTabControl>
            </c1:C1DockGroup>
            <c1:C1DockGroup>
                <c1:C1DockTabControl Dock="Top" DockWidth="500" DockHeight="500"
TabItemShape="Rounded">
                    <c1:C1DockTabItem Header="タブ 4">
                        <!--ここにコンテンツを配置-->
                    </c1:C1DockTabItem>
```

```
                    <c1:C1DockTabItem Header="タブ 5">
                        <!--ここにコンテンツを配置-->
                    </c1:C1DockTabItem>
                </c1:C1DockTabControl>
                <c1:C1DockTabControl>
                    <c1:C1DockTabItem Header="タブ 6">
                        <!--ここにコンテンツを配置-->
                    </c1:C1DockTabItem>
                </c1:C1DockTabControl>
            </c1:C1DockGroup>
            <c1:C1DockTabControl>
                <c1:C1DockTabItem Header="タブ 7" TabShape="Sloped">
                    <!--ここにコンテンツを配置-->
                </c1:C1DockTabItem>
                <c1:C1DockTabItem Header="タブ 8" TabShape="Sloped">
                    <!--ここにコンテンツを配置-->
                </c1:C1DockTabItem>
            </c1:C1DockTabControl>
</c1:C1DockControl>
```

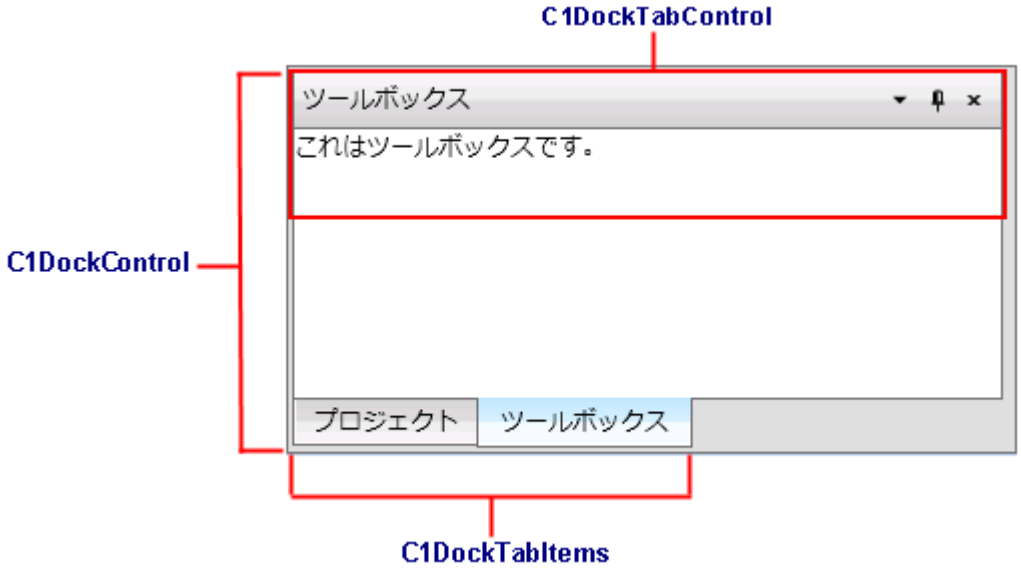
DockControl の使い方

DockControl for WPF/Sliverlight には**C1DockControl** コントロールが含まれています。これは、ウィンドウをドッキング、浮動化、またはタブ付きにできる簡単なコントロールです。以下のトピックでは、**C1DockControl** とそれに追加できる要素、ウィンドウをドッキングするためのオプション、ドッキングインジケータ、ドッキングゾーンについて詳しく説明します。

ドッキングコントロールの要素

C1DockControl はコンテナとして機能し、ここに、ウィンドウの整理に役立つ他の **DockControl for WPF/Silverlight** 要素を追加できます。

ページに**C1DockControl** を追加したら、**C1DockTabItem** を含む **C1DockTabControl** を追加できます。



C1DockTabControl には、次の3つのボタンがあります。

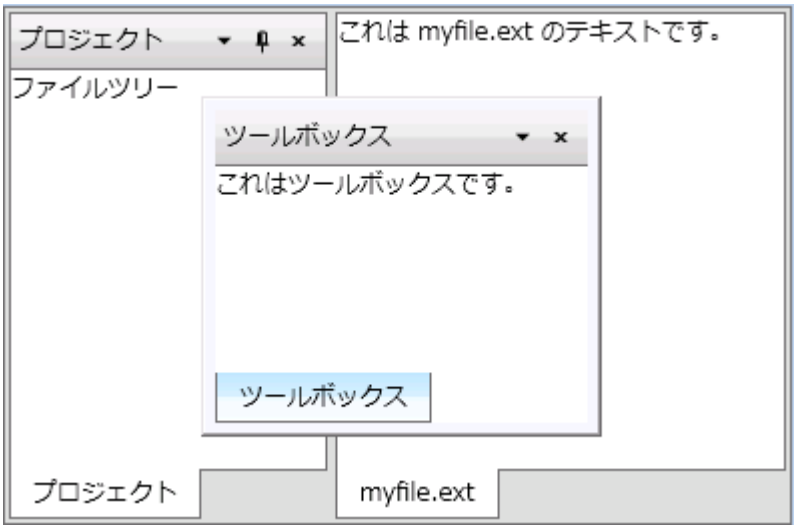
ドロップダウンリスト	DockMode オプション。Floating、Docked、Sliding、Hidden のオプションがあります。
押しピン	自動非表示機能を有効にします。**C1DockTabItems** は IDE の端に最小化されます。
非表示	**C1DockTabControl** を非表示にします。

ドッキングオプション

DockControl for WPF/Silverlight には、フローティング、ドッキング、自動的に隠す、非表示の4つの DockMode オプションがあります。以下の画像に、**C1DockTabControl** のドロップダウンリストから選択できるドッキングオプションを示します。

フローティング

[**フローティング**]をクリックすると、ウィンドウがアンドックされ、浮動化されて他のウィンドウの上に表示されます。

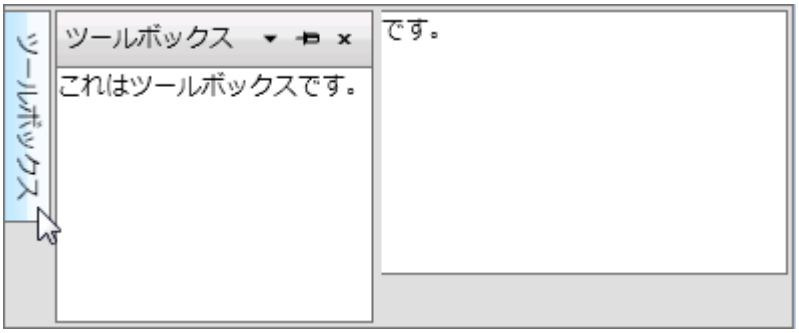


ドッキング

[ドッキング]をクリックすると、ウィンドウが他のウィンドウにドッキングされます。

自動的に隠す

[自動的に隠す]をクリックすると、**C1DockTabItem** が IDE の端に最小化されます。



非表示

[非表示]をクリックすると、**C1DockTabControl** が非表示になります。

ドッキングモードを設定するには

設計時に**DockMode** プロパティを使用することにより、ドッキングモードを設定できます。次の XAML マークアップは、ドッキングモードを **Floating** に設定します。

XAML

```
<c1:C1DockTabControl DockMode="Floating">
```

```
    <c1:C1DockTabItem Header="ツールボックス">
        <TextBlock Text="これはツールボックスです。" />
    </c1:C1DockTabItem>
</c1:C1DockTabControl>
```

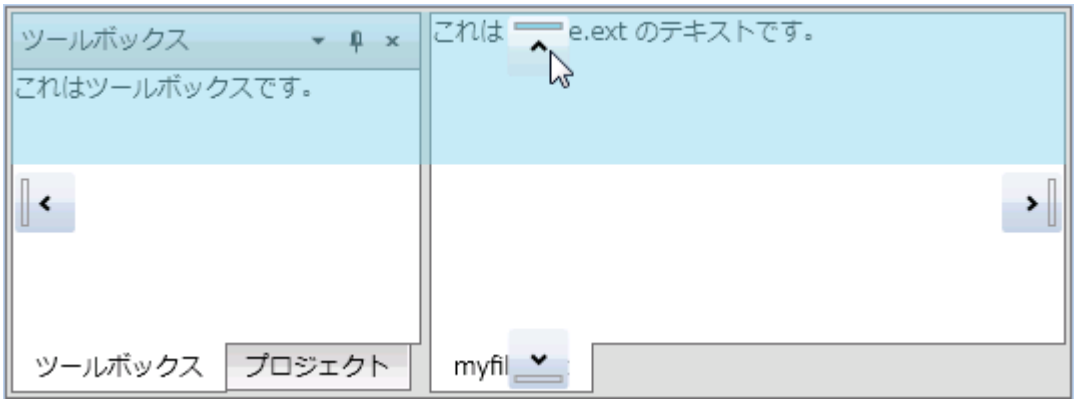
詳細については、「ドッキングモードの設定」を参照してください。

ひし形のドッキングガイドとゾーン

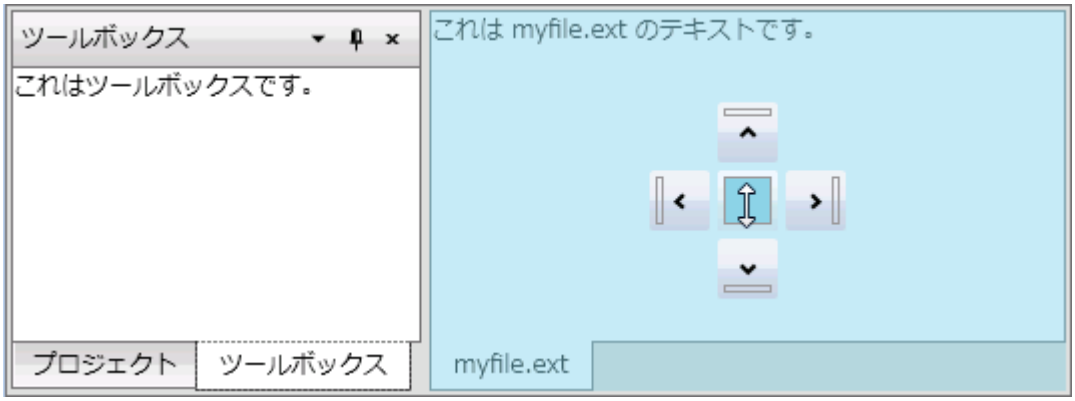
ひし形のドッキングガイドは、**C1DockControl** 要素をドッキングできる場所を示すために使用されます。

C1DockControl では4つのドッキングインジケータが提供されており、コントロールの上下左右の4辺のいずれかでドッキングできます。

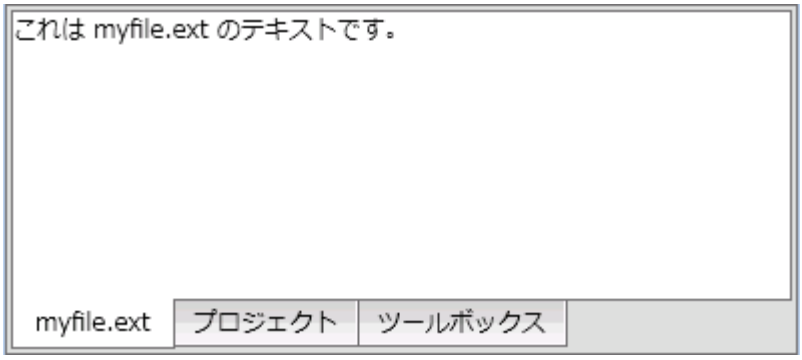
C1DockTabControl のヘッダーをドラッグすると、青いドッキングゾーンが表示されて、ウィンドウをどこにドッキングできるかがわかります。



C1DockTabControl が複数ある場合は、いずれかの **C1DockTabControl** のヘッダーを他の **C1DockTabControls** のヘッダー上にドラッグしてみてください。ひし形のドッキングガイドに上下左右および中央の合計5つのドッキングインジケータが表示されます。中央のドッキングインジケータを使用すると、**C1DockTabItems** を **C1DockTabControl** にマージできます。



上の画像例の状態で、中央のインジケータ上でマウスのボタンを放すと、myfile.ext **C1DockTabItem** と他の **C1DockTabItems** が最初の **C1DockTabControl** に表示されます。



レイアウトおよび外観

以下のトピックでは、**C1DockControl** 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、**DockControl for WPF/Sliverlight** 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

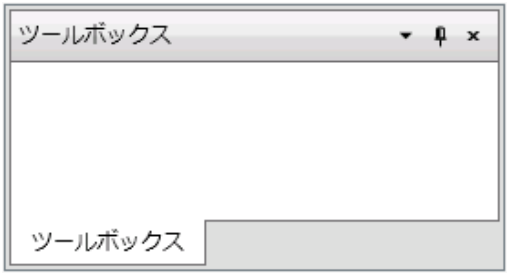
テンプレートにアクセスするには、Microsoft Expression Blend で、**C1DockControl** コントロールを選択し、メニューから**[テンプレートの編集]**を選択します。**[コピーの編集]**を選択して現在のテンプレートのコピーを作成して編集するか、**[空のテンプレートの作成]**を選択して新しい空のテンプレートを作成します。

> **メモ**:メニューから新しいテンプレートを作成する場合、テンプレートはそのテンプレートのプロパティに自動的にリンクされます。手作業でテンプレートの XAML を作成する場合は、作成したテンプレートに適切な Template プロパティをリンクする必要があります。

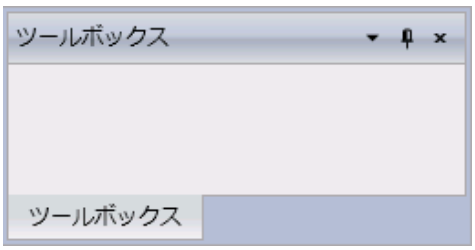
Template プロパティを使ってテンプレートをカスタマイズできます。

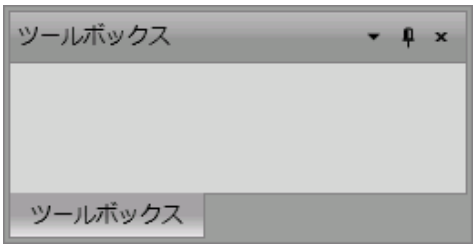
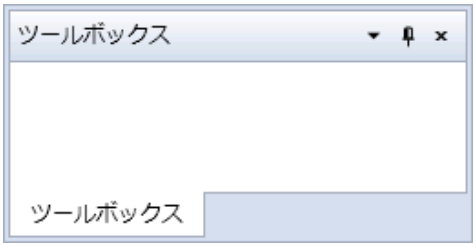
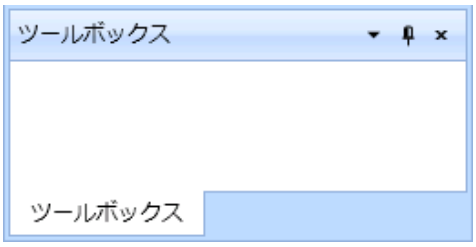
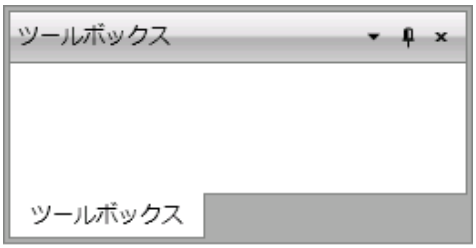
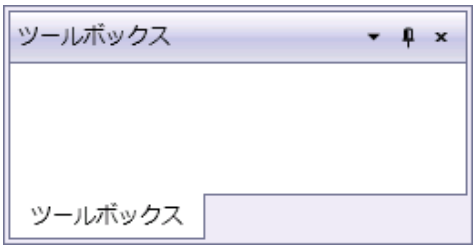
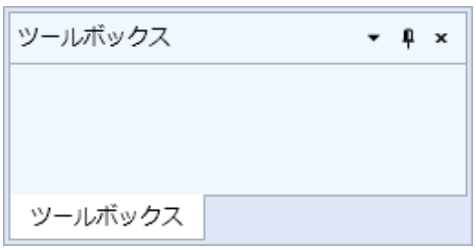
テーマ

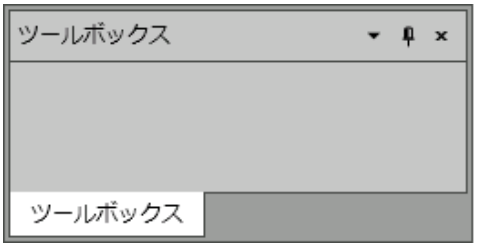
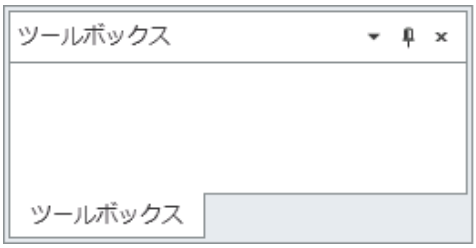
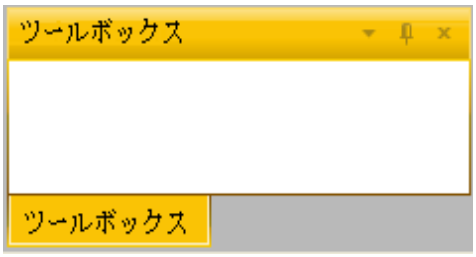
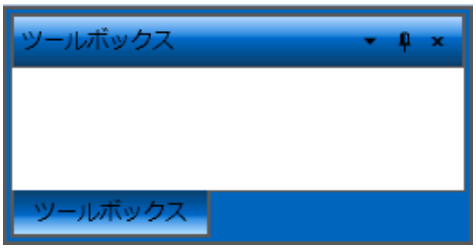
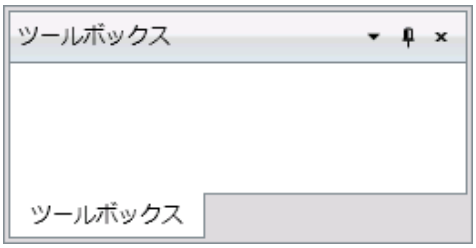
DockControl for WPF/Silverlight には、グリッドの外観をカスタマイズできるいくつかのテーマが組み込まれています。**C1DockTabControl** と C1DockTabItem を含む**C1DockControl** コントロールを初めてページに追加すると、次の図のように表示されます。



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。WPF のすべての組み込みテーマは、WPF Toolkit テーマに基づいています。以下に、組み込みテーマの説明と図を示します。以下の図では、選択状態のスタイルを示すために1つの行が選択されています。

テーマ名	テーマのプレビュー
C1ThemeBureauBlack	

C1ThemeExpressionDark	
C1ThemeExpressionLight	
C1Blue (WPF のみ)	
C1ThemeOffice2007Blue	
C1ThemeOffice2007Black	
C1ThemeOffice2007Silver	
C1ThemeOffice2010Blue	

C1ThemeOffice2010Black	
C1ThemeOffice2010Silver	
C1ThemeRainierOrange (Silverlight のみ)	
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。

VisualBasic

```
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。

VisualBasic

```
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。

> **メモ**:Silverlightの場合は、BureauBlack、ExpressionDark、ExpressionLight、RainierOrange、ShinyBlue、WhistlerBlue の6つのテーマが提供されています。

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。

ClearStyle プロパティ

ClearStyle プロパティ

DockControl for WPF/Sliverlight 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、**C1DockControl** 要素のスタイルを簡単に設定できます。次の表に、**C1DockControl** でサポートされているプロパティを一覧します。

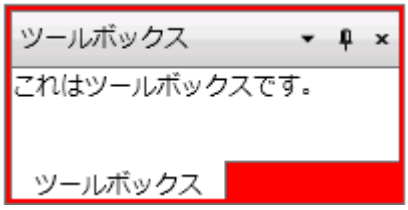
プロパティ	説明
Background	**C1DockControl** の塗りつぶしに使用される背景を取得または設定します。
MouseOverBrush	マウスポインタが置かれたコントロールを強調表示するために使用されるブラシを取得または設定します。
TabControlBackground	**C1DockTabControl** の背景色を取得または設定します。
TabControlForeground	**C1DockTabControl** の前景色またはテキストの色を取得または設定します。
TabStripBackground	**C1DockTabItem** の背景色を取得または設定します。
TabStripForeground	**C1DockTabItem** の前景色またはテキストの色を取得または設定します。

これらのプロパティを設定することで、**C1DockControl** の外観を完全に変更できます。たとえば、**Background** プロパティを **Red** に設定する場合の XAML マークアップは次のようになります。

XAML

```
<c1:C1DockControl Height="100" Background="Red" HorizontalAlignment="Left"
Margin="176,100,0,0" Name="c1DockControl1" VerticalAlignment="Top" Width="200">
  <c1:C1DockTabControl Height="100" HorizontalAlignment="Left"
Name="c1DockTabControl1" VerticalAlignment="Top" Width="200">
    <c1:C1DockTabItem Header="ツールボックス" HorizontalAlignment="Left"
Name="c1DockTabItem1" VerticalAlignment="Top" >
     <TextBlock Text="これはツールボックスです。" />
        </c1:C1DockTabItem>
        </c1:C1DockTabControl>
      </c1:C1DockControl>
```

C1DockControl は次のようになります。

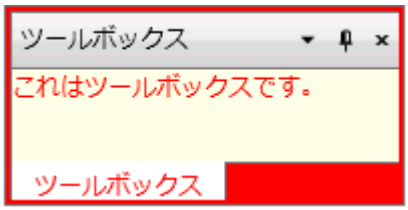


その他のプロパティも試して、**C1DockControl** 要素の外観をすばやく変更してみてください。たとえば、次の XAML は、**Background**、**TabStripForeground**、**TabControlBackground** の各プロパティを設定します。

XAML

```
<c1:C1DockControl Height="100" Background="Red" TabStripForeground="Red"
TabControlBackground="LightYellow" HorizontalAlignment="Left" Margin="176,100,0,0"
Name="c1DockControl1" VerticalAlignment="Top" Width="200">
    <c1:C1DockTabControl Height="100" HorizontalAlignment="Left"
Name="c1DockTabControl1" VerticalAlignment="Top" Width="200">
      <c1:C1DockTabItem Header="ツールボックス" HorizontalAlignment="Left"
Name="c1DockTabItem1" VerticalAlignment="Top" />
     <TextBlock Text="これはツールボックスです。" />
       </c1:C1DockTabItem>
       </c1:C1DockTabControl>
  </c1:C1DockControl>
```

C1DockControl は次のようになります。



タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio でのプログラミングに精通しており、**C1DockControl** 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。**DockControl for WPF/Sliverlight** 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、**DockControl for WPF/Sliverlight** 製品を使用して特定のタスクを行うための方法を提供します。

また、タスク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。

ドッキングモードの設定

設計時に **DockMode** プロパティを使用することにより、ドッキングモードを設定できます。

ドッキングモードを設定するには、次の手順に従います。

1. Visual Studio で、.xaml ページを開きます。
2. <Grid> </Grid>タグの間にカーソルを置きます。
3. ツールボックスで、[**C1DockControl**] アイコンをダブルクリックして、このコントロールをプロジェクトに追加します。
4. <c1:C1DockControl> タグと </c1:C1DockControl> タグの間にカーソルを置きます。
5. ツールボックスで、[**C1DockTabControl**] アイコンをダブルクリックして、このコントロールをプロジェクトに追加します。
6. **DockMode** プロパティを **Sliding** に設定します。XAML マークアップは次のようになります。

XAML

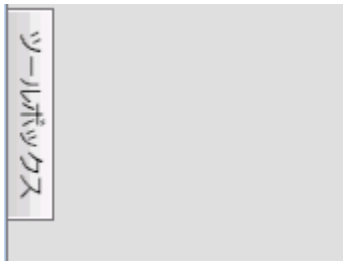
```
        <c1:C1DockControl>
        <c1:C1DockTabControl DockMode="Sliding"></c1:C1DockTabControl>
    </c1:C1DockControl>
```

7. <c1:C1DockTabControl> タグと </c1:C1DockTabControl>タグの間にカーソルを置きます。
8. ツールボックスで、[**C1DockTabItem**]アイコンをダブルクリックしてコントロールをプロジェクトに追加し、**C1DockTabItem.Header** プロパティを「**ツールボックス**」に設定します。XAML は、次のようになります。

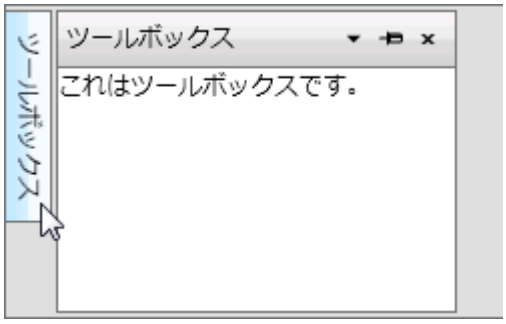
XAML

```
            <c1:C1DockControl>
            <c1:C1DockTabControl DockMode="Sliding">
                <c1:C1DockTabItem Header="ツールボックス"></c1:C1DockTabItem>
            </c1:C1DockTabControl>
    </c1:C1DockControl>
```

9. プロジェクトを実行します。**C1DockControl** は、次の図のようになります。



10. [**ツールボックス**]タブをクリックして、ウィンドウをスライドして表示させます。



ドッキング位置を無効にする方法

PickerLoading イベント内で、ユーザーが特定の方向にドッキングできないようにすることができます。たとえば、**C1DockControl** の左外側へのドッキングを無効にするには、次の手順に従います。

1. **C1DockControl** といくつかの子 **C1DockTabControl** をページに追加します。
2. **C1DockControl** の **PickerLoading** イベントをサブスクライブします。

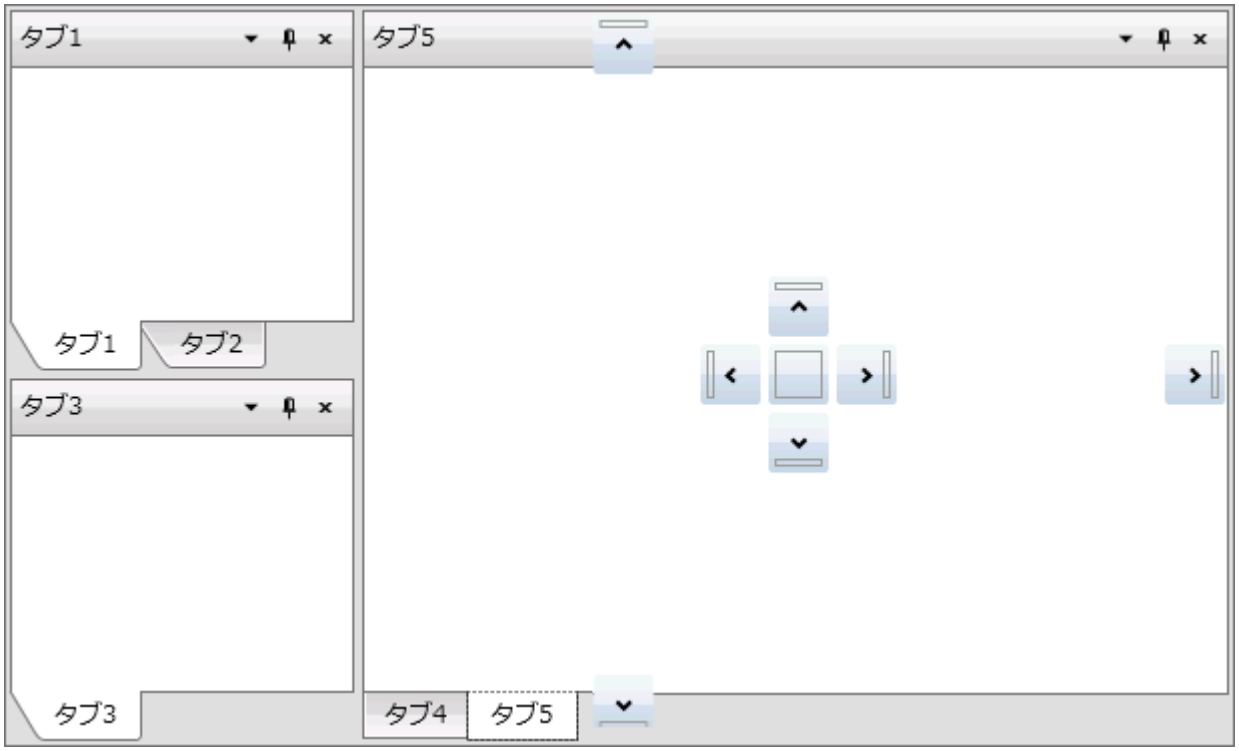
XAML

```
   <c1:C1DockControl Name="dockControl1"
PickerLoading="dockControl1_PickerLoading" />
```

3. コードで、ドッキングピッカーのいくつかの部分が表示されないように、任意の数の **PickerLoadingEventArgs** プロパティを設定します。たとえば、左外側のドッキングを無効にするには、**e.ShowLeftOuterPart** プロパティを **False** に設定します。

C#

```
private void dockControl1_PickerLoading(object sender, PickerLoadingEventArgs e)
{
    // 左外側のドッキングを無効にします
    e.ShowLeftOuterPart = false;
}
```



条件付きドッキング

特定の **C1DockTabControl** または **C1DockTabItem** に対して条件付きのドッキングを行うには、**Source** プロパティと **Target** プロパティを使用します。たとえば、**c1DockTabControl2** の内側に **c1DockTabItem1** をドッキングできないようにするには、次のコードを記述します。

XAML

```
private void dockControl1_PickerLoading(object sender, PickerLoadingEventArgs e)
{
    if (e.Source == c1DockTabItem1 && e.Target == c1DockTabControl2)
    {
        // ドッキングをすべて無効にします
        e.ShowBottomInnerPart = false;
        e.ShowLeftInnerPart = false;
        e.ShowRightInnerPart = false;
        e.ShowTopInnerPart = false;
        e.ShowOverInnerPart = false;
    }
}
```