DateTimeEditors for WPF/Silverlight

2018.02.20 更新

グレープシティ株式会社

目次

製品の概要

DateTime Editors for WPF/Silverlight を使って DateTime 情報を表示、編集、および検証します。**C1DateTimePicker** コントロールは、単体で日付と時刻の値を選択できる直観的な UI を提供します。**C1TimeEditor** コントロールは、時刻値のための簡単なマスクエディタを提供します。ユーザーは、スピンボタンまたはキーボードの矢印キーを使用するか、フィールドに値を入力して、日付と時刻の値を編集できます。

> **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 Studio for WPF/Silverlight のヘルプ

はじめに

ComponentOne Studio 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 Studio for WPF/Silverlight ユーザーガイド」を参照してください。

主な特長

DateTimeEditors 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、**DateTimeEditors for WPF/Silverlight** を最大限に活用してください。

- **複数の表示バージョン**
  編集モードとして、**DateTime(デフォルト)**、**Date**、**Time** のいずれかを選択します。日付書式として、**Short** と **Long** を含む設定済みの書式から選択します。時刻書式として、**ShortTime**、**LongTime**、および **TimeSpan** を含む設定済みの書式から選択します。
- **スピンボタンのサポート**
  C1DateTimePicker コントロールおよび **C1TimeEditor** コントロールでは、スピン(上/下)ボタンを使って日付と時刻を選択できます。
- **幅広いカルチャ**
  日付と時刻の書式のカルチャ設定を定義します。
- **null 値のサポート**
  C1DateTimePicker コントロール、**C1DatePicker** コントロールおよび **C1TimeEditor** コントロールでは、null 値をデフォルトで入力できます。これは、**AllowNull** プロパティを **False** に設定することによって無効にできます。

XAML クイックリファレンス

このトピックでは、**C1DateTimePicker** コントロールの作成に使用される XAML の概要を提供します。このセクションの XAML マークアップは、**C1DateTimePicker** コントロールを作成し、日付/時刻と日付書式を設定する方法を示します。

XAML

```
<c1datetime:C1DateTimePicker DateTime="1/17/2010 11:04 AM" DateFormat="Long" />
```

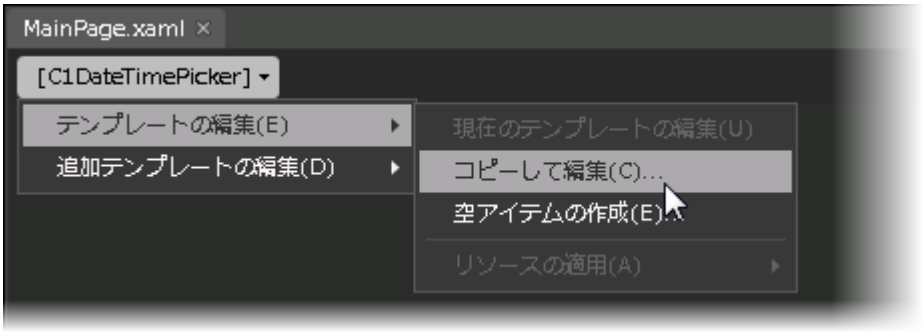
テンプレート (Silverlightのみ)

テンプレート

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」コントロールだからです。Silverlight アプリケーションで独自のユーザーインターフェイス（UI）、つまり「外観」を設計するのと同様に、**DateTimeEditors for Silverlight** によって管理されるデータにも独自の UI を提供できます。Extensible Application Markup Language（XAML。「ザムル」と発音する）は、コードを記述することなく独自の UI を設計するための簡単な方法を提供する XML ベースの宣言型言語です。

テンプレートへのアクセス

テンプレートにアクセスするには、Expression Blend で、C1DateTimePicker、C1DatePicker または C1TimeEditor 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。



メモ:メニューを使っ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。

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:このトピックの内容は、ComponentOne Studio for Silverlight にのみ適用されます。

C1DateTimePicker コントロール

DateTimePicker for WPF/Silverlight を使用して、日付と時刻の情報を交換します。この機能は、日付と時刻の値または時刻のみの値を選択するための簡単で直感的な UI を提供します。ユーザーは、スピンボタンまたはキーボードの矢印キーを使用するか、フィールドに値を入力して、日付と時刻を選択できます。

C1DateTimePicker クイックスタート

このクイックスタートは、**C1DateTimePicker** 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 **C1DateTimePicker** コントロールを追加し、**C1DateTimePicker** コントロールをカスタマイズします。

手順 1：C1DateTimePicker コントロールを含むアプリケーションの作成

この手順では、最初に Visual Studio で **C1DateTimePicker** を使用する WPF/Silverlight アプリケーションを作成します。

次の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新しいプロジェクト]**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**[WPF アプリケーション]**または**[Silverlight アプリケーション]**を選択します。
3. プロジェクトの**名前と場所**を入力し、**[OK]**をクリックして新しいアプリケーションを作成します。
4. WPF プロジェクトを作成する場合、ツールボックスで、**C1DateTimePicker** アイコンをダブルクリックして、**C1DateTimePicker** コントロールを WPF アプリケーションに追加します。

   Silverlight プロジェクトを作成する場合、以下の手順に従います。

   a. プロジェクトの XAML ウィンドウで、タグの **DesignWidth="400" DesignHeight="300"** を **DesignWidth="Auto" DesignHeight="Auto"** に変更して、**UserControl** をサイズ変更します。次のようになります。

   XAML

```
<UserControl x:Class="SilverlightApplication1.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="Auto" d:DesignHeight="Auto">
```

   これで、**UserControl** は、中に置かれた内容に合わせてサイズ変更されるようになります。

   b. 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

   c. ツールボックスに移動し、**C1DateTimePicker** アイコンをダブルクリックして、コントロールをグリッドに追加します。XAML マークアップは次のようになります。

   XAML

```
<Grid x:Name="LayoutRoot">
```

```
<c1:C1DateTimePicker></c1:C1DateTimePicker>
</Grid>
```

これで、**C1DateTimePicker** コントロールを含む WPF/Silverlight アプリケーションを作成できました。次の手順では、**C1DateTimePicker** コントロールにビデオコンテンツを追加します。

手順 2:コントロールのカスタマイズ

前の手順では、**C1DateTimePicker** コントロールを使って WPF/Silverlight アプリケーションを作成しました。この手順では、コントロールの外観を変更します。

WPF の場合

C1DateTimePicker コントロールを選択し、[プロパティ]ウィンドウで次のプロパティを設定します。

- **Height** プロパティを「30」に設定して、コントロールの高さを設定します。
- **Width** プロパティを「300」に設定して、コントロールの幅を設定します。
- **TimeFormat** プロパティを **ShortTime** に設定して、時刻書式を時間と分の領域だけで構成される短い書式に変更します。
- **DateFormat** プロパティを **Long** に設定して、日付書式を曜日を含む長い書式に変更します。
- **SelectionBackground** プロパティを **LimeGreen** に設定して、コントロール内の選択された領域の色を変更します。
- **FirstDayOfWeek** プロパティを **Wednesday** に設定して、ドロップダウンカレンダーの週の最初の曜日を水曜日に変更します。

Silverlight の場合

次の手順に従います。

1. <c1:C1DateTimePicker> タグに **Height="30"** を追加し、コントロールの高さを指定します。XAML マークアップは次のようになります。

```
XAML
<c1:C1DateTimePicker Height="30">
```

2. <c1datetime:C1DateTimePicker> タグに **Width="300"** を追加し、コントロールの幅を指定します。XAML マークアップは次のようになります。

```
XAML
<c1:C1DateTimePicker Height="30" Width="300">
```

3. <c1datetime:C1DateTimePicker> タグに **TimeFormat="ShortTime"** を追加し、時刻書式を時間と分の領域だけで構成される短い書式に変更します。XAML マークアップは次のようになります。

```
XAML
<c1:C1DateTimePicker Height="30" Width="300" TimeFormat="ShortTime">
```

4. <c1datetime:C1DateTimePicker> タグに **DateFormat="Long"** を追加し、日付書式を曜日を含む長い書式に変更します。XAML マークアップは次のようになります。

XAML

```
<c1:C1DateTimePicker Height="30" Width="300" TimeFormat="ShortTime"
DateFormat="Long">
```

5. <c1datetime:C1DateTimePicker> タグに **SelectionBackground="LimeGreen"** を追加します。これにより、C1DateTimePicker コントロールの選択した領域の色が変更されます。XAML マークアップは次のようになります。

XAML

```
<c1:C1DateTimePicker Height="30" Width="300" TimeFormat="ShortTime"
DateFormat="Long"SelectionBackground="LimeGreen">
```

6. <c1datetime:C1DateTimePicker> タグに **FirstDayOfWeek="Wednesday"** を追加します。これにより、ドロップダウンカレンダーの週の最初の曜日が水曜日に変更されます。XAML マークアップは次のようになります。

XAML

```
<c1:C1DateTimePicker Height="30" Width="300" TimeFormat="ShortTime"
DateFormat="Long" SelectionBackground="LimeGreen" FirstDayOfWeek="Wednesday">
```

この手順では、**C1DateTimePicker** コントロールの外観をカスタマイズしました。次の手順では、プロジェクトを実行し、コントロールの機能を確認します。

手順 3:プロジェクトの実行

前の手順では、**C1DateTimePicker** コントロールをカスタマイズしました。この手順では、プロジェクトを実行し、コントロールの実行時機能をいくつか確認します。

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、アプリケーションを実行します。ビデオコンテンツが自動的に再生され、次のようなアプリケーションが表示されることを確認します。

2. カーソルを使用して、日付ピッカーの領域を強調表示します。選択範囲は次のようになります。

3. 時刻ピッカーのドロップダウン矢印をクリックしてカレンダーを表示し、カレンダーの週が水曜日から始まっていることを確認します。

おめでとうございます!

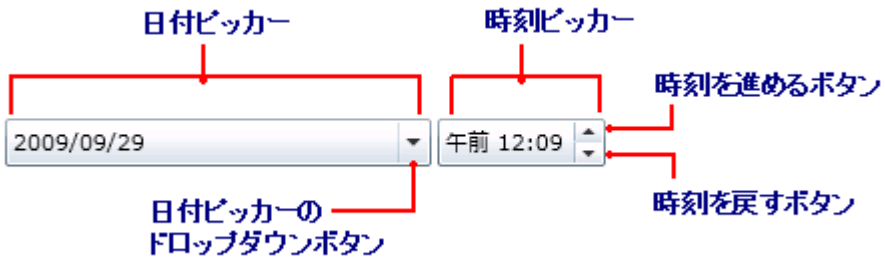
これで、**C1DateTimePicker** クイックスタートは終了です。このクイックスタートでは、**C1DateTimePicker** コントロールを含む WPF/Silverlight アプリケーションを作成し、コントロールの外観を変更しました。このクイックスタートは終了したので、次に「C1DateTimePicker の使い方」または「タスク別ヘルプ 」のトピックを参照することをお勧めします。

C1DateTimePicker の使い方

以下のトピックでは、**C1DateTimePicker** コントロールのすべての要素といくつかの機能を紹介します。

C1DateTimePicker の要素

DateTimeEditors for WPF/Silverlight には、**C1DateTimePicker** コントロールが含まれています。これは、デフォルトで日付ピッカーと時刻ピッカーの両方を提供するシンプルなコントロールです。XAML ウィンドウに追加された **C1DateTimePicker** コントロールは、完全な機能を備えた日付時刻ピッカーになります。デフォルトでは、コントロールのインターフェイスは次の図のように表示されます。



C1DateTimePicker コントロールは、次の要素で構成されます。

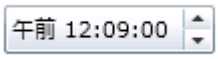
- **日付ピッカー**
  日付ピッカー要素は、日付フィールドとカレンダードロップダウンボタンで構成されます。日付は、数値を入力するか、またはカレンダーから日付を選択して設定できます。
- **時刻ピッカー**
  時刻ピッカー要素は、時刻フィールド、時刻を進めるボタン、および時刻を戻すボタンで構成されます。時刻は、数値を入力するか、またはボタンをクリックして設定できます。
- **日付ピッカーのドロップダウンボタン**
  日付ピッカーのドロップダウンボタンをクリックするとカレンダーが開くので、そこから日付ピッカーの日付を選択できます。

- **時刻を進めるボタン**
  時刻を進めるボタンを使用すると、時刻ピッカーに表示する時刻を進めることができます。進めるボタンをクリックすると、時刻が1分進みます。

- **時刻を戻すボタン**
  時刻を戻すボタンを使用すると、時刻ピッカーに表示する時刻を戻すことができます。戻すボタンをクリックすると、時刻が1分戻ります。

編集モード

デフォルトでは、ページに配置された **C1DateTimePicker** コントロールには日付ピッカーと時刻ピッカーの両方が表示されます。表示されるピッカーは、**EditMode** プロパティを **Date**、**Time**、または **DateTime** に設定することによって変更できます。日付ピッカーのみを表示する場合は **EditMode** プロパティを **Date** に、時刻ピッカーのみを表示する場合は **EditMode** プロパティを **Time** に、日付ピッカーと時刻ピッカーの両方を表示する場合は **EditMode** プロパティを **DateTime** に設定します。次の表に、それぞれのエディタモードを示します。

エディタモード	結果
Date	
Time	
DateTime	

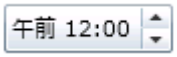
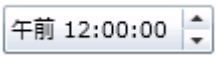
日付書式

DateFormat プロパティを使用すると、日付ピッカーを表示するときの書式を設定できます。**DateFormat** プロパティは、**Short** または **Long** に設定できます。次の表に、2つの日付書式を示します。

日付書式	結果
Short（デフォルト）	
Long	

時刻書式

TimeFormat プロパティを使用すると、日付ピッカーを表示するときの書式を設定できます。**TimeFormat** プロパティは、**ShortTime** または **LongTime** に設定できます。次の表に、2つの日付書式を示します。

時刻書式	結果
ShortTime	
LongTime（デフォルト）	

レイアウトおよび外観

以下のトピックでは、**C1DateTimePicker** 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/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。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。

ComponentOne ClearStyle 技術

DateTimePicker 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、グリッド全体のスタイルを簡単に設定できます。

次の表に、**C1DateTimePicker** コントロールのブラシのプロパティの概要を示します。

ClearStyle プロパティ	説明
Background	コントロールの背景のブラシを取得または設定します。
ButtonBackground	ボタンの背景色のブラシを取得または設定します。
ButtonForeground	ボタンの前景色のブラシを取得または設定します。
MouseOverBrush	マウスポインタが置かれたボタンを強調表示するために使用される System.Windows.Media.Brush を取得または設定します。
PressedBrush	クリックされたボタンを強調表示するために使用される System.Windows.Media.Brush を取得または設定します。

いくつかのプロパティを設定することで、**C1DateTimePicker** コントロールの外観を完全に変更できます。たとえば、**ButtonBackground** は、**C1DateTimePicker** コントロールのドロップダウン矢印の背景を設定します。**ButtonBackground** プロパティを "#FFC500FF" に設定すると、**C1DateTimePicker** コントロールは次のようになります。

外観プロパティ

C1DateTimePicker コントロールには、コントロールの外観をカスタマイズするための複数のプロパティが含まれます。コントロールに表示されるテキストの外観を変更したり、コントロールのグラフィック要素をカスタマイズすることができます。以下のトピックでは、これらの外観プロパティの一部について説明します。

テキストのプロパティ

次のプロパティを使用して、**C1DateTimePicker** コントロールのテキストの外観をカスタマイズできます。

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

色のプロパティ

次のプロパティを使用して、コントロール自体に使用される色をカスタマイズできます。

プロパティ	説明
Background	コントロールの背景を描画するブラシを取得または設定します。これは依存プロパティです。
Foreground	前景色を描画するブラシを取得または設定します。これは依存プロパティです。

境界線のプロパティ

次のプロパティを使用して、コントロールの境界線をカスタマイズできます。

プロパティ	説明
BorderBrush	コントロールの境界線の背景を描画するブラシを取得または設定します。これは依存プロパティです。
BorderThickness	コントロールの境界線の太さを取得または設定します。これは依存プロパティです。

サイズのプロパティ

次のプロパティを使用して、**C1DateTimePicker** コントロールのサイズをカスタマイズできます。

プロパティ	説明
ActualHeight	この要素のレンダリングされた高さを取得または設定します。これは依存プロパティです。
ActualWidth	この要素のレンダリングされた幅を取得または設定します。これは依存プロパティです。
Height	要素の推奨高さを取得または設定します。これは依存プロパティです。
MaxHeight	要素の最大高さ制約を取得または設定します。これは依存プロパティです。
MaxWidth	要素の最大幅制約を取得または設定します。これは依存プロパティです。
MinHeight	要素の最小高さ制約を取得または設定します。これは依存プロパティです。
MinWidth	要素の最小幅制約を取得または設定します。これは依存プロパティです。
Width	要素の幅を取得または設定します。これは依存プロパティです。

テーマ

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。

C1DateTimePicker コントロールをプロジェクトに追加すると、次の図に示すように、コントロールはデフォルトの青色のテーマで表示されます。

C1DateTimePicker コントロールには、WPF の13のつのテーマ（BureauBlack、ExpressionDark、ExpressionLight、Office2007Black、Office2007Blue、Office2007Silver、Office2010Black、Office2010Blue、Office2010Silver、RainierOrange、ShinyBlue、WhistlerBlue）の中から1つテーマを設定できます。次の表に、テーマごとのサンプルを示します。

完全なテーマ名	外観
C1ThemeBureauBlack	
C1ThemeExpressionDark	
C1Blue (WPF のみ)	

C1ThemeExpressionLight	
C1ThemeOffice2007Black	
C1ThemeOffice2007Blue	
C1ThemeOffice2007Silver	

C1ThemeOffice2010Black	
C1ThemeOffice2010Blue	
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。

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、**C1DateTimePicker** 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。**C1DateTimePicker** コントロール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、**C1DateTimePicker** 製品を使っ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。

Null 値の許可

デフォルトでは、**C1DateTimePicker** コントロールは null 値の入力を許可しませんが、**AllowNull** プロパティを **True** に設定すると、コントロールに null 値の受け入れを強制できます。このトピックでは、デザイナ、XAML、およびコードで、**AllowNull** プロパティを **True** に設定する方法について説明します。

デザイナの場合

次の手順に従います。

1. **C1DateTimePicker** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**[AllowNull]** チェックボックスをオンにします。

XAML の場合

null 値を許可するには、**AllowNull="True"** を <my:C1DateTimePicker>タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1DateTimePicker AllowNull="True"/>
```

コードの場合

次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1DateTimePicker1.AllowNull = True
```

C#

```
c1DateTimePicker1.AllowNull = true;
```

3. プロジェクトを実行します。

> **メモ:**実行時に null 値を設定するには、コントロールの **DatePicker** 部分にあるテキストをすべてクリアし、[Enter]キーを押すか、コントロールの **TimeEditor** 部分にフォーカスを移動します。

編集モードの選択

デフォルトでは、**C1DateTimePicker** コントロールには日付ピッカーと時刻ピッカーの両方が表示されますが、日付ピッカーまたは時刻ピッカーのみを表示することもできます。このトピックでは、デザイナ、XAML、およびコードで、エディタモードを変更する方法について説明します。

デザイナの場合

編集モードを変更するには、次の手順に従います。

1. **C1DateTimePicker** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**EditMode** のドロップダウン矢印をクリックし、リストからモードを選択します。この例では、**[Date]**を選択します。

XAML の場合

編集モードを変更するには、EditMode="Date" を <my:C1DateTimePicker> タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1DateTimePicker EditMode="Date">
```

コードの場合

編集モードを変更するには、次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. 次の名前空間をインポートします。

WPF

Visula Basic

```
Imports C1.WPF.DateTimeEditors
```

C#

```
using C1.WPF.DateTimeEditors;
```

Sliverlight

Visual Basic

```
Imports C1.Silverlight.DateTimeEditors
```

C#

```
using C1.Silverlight.DateTimeEditors;
```

3. **InitializeComponent()** メソッドの下に次のコードを追加します。

WPF

Visual Basic

```
C1DateTimePicker1.EditMode = C1DateTimePickerEditMode.Date
```

C#

```
c1DateTimePicker1.EditMode = C1DateTimePickerEditMode.Date;
```

Silverlight

Visual Basic

```
C1DateTimePicker1.EditMode = C1DateTimePickerEditMode.Date
```

C#

```
C1DateTimePicker1.EditMode = C1DateTimePickerEditMode.Date;
```

4. プロジェクトを実行します。

このトピックの作業結果

このトピックでは、**EditMode** を **Date** に変更することにより、**C1DateTimePicker** コントロールから時刻ピッカーを除去します。このトピックの結果は、次の画像のようになります。

時刻書式の選択

デフォルトでは、**C1DateTimePicker** コントロールの時刻ピッカーは、秒を含む長い書式で時刻を表示しますが、時刻を短い書式で表示することもできます。このトピックでは、デザイナ、XAML、およびコードで、時刻書式を変更する方法について説明します。

デザイナの場合

時刻書式を変更するには、次の手順に従います。

1. **C1DateTimePicker** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**TimeFormat** のドロップダウン矢印をクリックし、リストからモードを選択します。この例では、**[ShortTime]**を選択します。

XAML の場合

時刻書式を変更するには、**TimeFormat="ShortTime"** を <my:C1DateTimePicker> タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1DateTimePicker TimeFormat="ShortTime">
```

コードの場合

時刻書式を変更するには、次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. 次の名前空間をインポートします。

WPF

Visual Basic

```
Imports C1.WPF.DateTimeEditors
```

C#

```
using C1.WPF.DateTimeEditors;
```

Silverlight

Visual Basic

```
Imports C1.Silverlight.DateTimeEditors
```

C#

```
using C1.Silverlight.DateTimeEditors;
```

3. **InitializeComponent()** メソッドの下に次のコードを追加します。

WPF

Visual Basic

```
C1DateTimePicker1.TimeFormat = C1TimeEditorFormat.ShortTime
```

C#

```
c1DateTimePicker1.TimeFormat = C1TimeEditorFormat.ShortTime;
```

Silverlight

Visual Basic

```
C1DateTimePicker1.TimeFormat = C1TimeEditorFormat.ShortTime
```

C#

```
C1DateTimePicker1.TimeFormat = C1TimeEditorFormat.ShortTime;
```

4. プロジェクトを実行します。

このトピックの作業結果

このトピックでは、**TimeFormat** を **ShortTime** に設定して、短い書式で時刻を表示しました。最終的な結果は、次の画像のようになります。

日付書式の選択

デフォルトでは、**C1DateTimePicker** コントロールの日付ピッカーは、短い書式で日付を表示しますが、日付を長い書式で表示することもできます。このトピックでは、デザイナ、XAML、およびコードで、日付書式を変更する方法について説明します。

デザイナの場合

日付書式を変更するには、次の手順に従います。

1. **C1DateTimePicker** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**DateFormat** のドロップダウン矢印をクリックし、リストからモードを選択します。この例では、**[Long]**を選択します。

XAML の場合

日付書式を変更するには、**DateFormat="Long"** を <my:C1DateTimePicker> タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1DateTimePicker DateFormat="Long">
```

コードの場合

日付書式を変更するには、次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1DateTimePicker1.DateFormat = Microsoft.Windows.Controls.DatePickerFormat.Long
```

C#

```
c1DateTimePicker1.DateFormat = Microsoft.Windows.Controls.DatePickerFormat.Long;
```

3. プロジェクトを実行します。

このトピックの作業結果

このトピックでは、**DateFormat** を **Long** に設定して、日付を長い書式で表示しました。最終的な結果は、次の画像のようになります。

カレンダーの最小日と最大日の設定

デザイナ、XAML、またはコードで **MinimumDate** プロパティと **MaximumDate** プロパティを設定することにより、カレンダーの日付範囲を変更できます。

> **メモ:MinDate** プロパティおよび **MaxDate** プロパティを XAML で文字列として設定しないでください。この値を文字列から解析する操作は、カルチャに依存します。現在のカルチャで値を設定したが、ユーザーが別のカルチャを使用している場合は、サイトを読み込む際に **XamlParseException** を受け取る可能性があります。最善の方法は、これらの値をコードまたはデータ連結を介して設定することです。

デザイナの場合

次の手順に従います。

1. **C1DateTimePicker** コントロールをクリックして選択します。
2. [プロパティ]ウィンドウで、次のプロパティを設定します。
   - **MinDate** プロパティを "01/01/2008" に設定します。
   - **MaxDate** プロパティを "12/31/2012" に設定します。
3. プログラムを実行します。
4. 日付ピッカーのドロップダウンボタンをクリックしてカレンダーを表示します。

5. 進むボタン( )をクリックして、それ以上進むことができないところまでカレンダーを前に進めます。カレンダーが 2012 年 12 月で止まることを確認します。
6. 戻るボタン( )をクリックして、それ以上戻ることができないところまでカレンダーを後に進めます。カレンダーが 2008 年1月で止まることを確認します。

XAML の場合

次の手順に従います。

1. **MinDate="2008-01-01"** と **MaxDate="2012-12-31"** を <my:C1DateTimePicker> タグに追加します。マークアップは、次のようになります。

XAML

```
<my:C1DateTimePicker Height="26" Name="c1DateTimePicker1" MinDate="2008-01-01"
MaxDate="2012-12-31" />
```

2. プログラムを実行します。
3. 日付ピッカーのドロップダウンボタンをクリックしてカレンダーを表示します。

4. 進むボタン( )をクリックして、それ以上進むことができないところまでカレンダーを前に進めます。カレンダーが 2012 年 12 月で止まることを確認します。
5. 戻るボタン( )をクリックして、それ以上戻ることができないところまでカレンダーを後に進めます。カレンダーが 2008 年1月で止まることを確認します。

コードの場合

次の手順に従います。

1. C1DateTimePicker コントロールをクリックして選択します。
2. [プロパティ]ウィンドウで、次のプロパティを設定します。
3. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
'最小日を設定します
C1DateTimePicker1.MinDate = new DateTime(2008, 01, 01)
'最大日を設定します
C1DateTimePicker1.MaxDate = new DateTime(2012, 12, 31)
```

C#

```
//最小日を設定します
c1DateTimePicker1.MinDate = new DateTime(2008, 01, 01);
//最大日を設定します
c1DateTimePicker1.MaxDate = new DateTime(2012, 12, 31);
```

4. プログラムを実行します。

5. 日付ピッカーのドロップダウンボタンをクリックしてカレンダーを表示します。
6. 進むボタン( )をクリックして、それ以上進むことができないところまでカレンダーを前に進めます。カレンダーが 2012 年 12 月で止まることを確認します。
7. 戻るボタン( )をクリックして、それ以上戻ることができないところまでカレンダーを後に進めます。カレンダーが 2008

年1月で止まることを確認します。

日付と時刻の指定

C1DateTimePicker コントロールの時刻と日付を指定するには、XAML またはコードを使って **DateTime** プロパティを設定します。

> **メモ:DateTime** プロパティを XAML で設定しないでください。この値を文字列から解析する操作は、カルチャに依存します。現在のカルチャで値を設定したが、ユーザーが別のカルチャを使用している場合は、サイトを読み込む際に XamlParseException を受け取る可能性があります。最善の方法は、これらの値をコードまたはデータ連結を介して設定することです。

XAML の場合

次の手順に従います。

1. DateTime="1/17/2010 11:04 AM" を <my:C1DateTimePicker> タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1DateTimePicker DateTime="1/17/2010 11:04 AM" />
```

2. 2.プロジェクトを実行し、**C1DateTimePicker** コントロールに日付と時刻が「2010/01/17 午前 11:04:00」と表示されることを確認します。

コードの場合

次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1DateTimePicker1.DateTime = New DateTime(2010, 1, 17, 11, 04, 0)
```

C#

```
c1DateTimePicker1.DateTime = new DateTime(2010, 1, 17, 11, 04, 0);
```

3. プロジェクトを実行し、C1DateTimePicker コントロールに日付と時刻が「2010/01/17 午前 11:04:00」と表示されることを確認します。

このトピックの作業結果

このトピックの結果は、次の図のようになります。

テーマを使用する (Silverlightのみ)

C1DateTimePicker コントロールには、デフォルトで明るい青色のテーマが設定されていますが、全部で 複数 のテーマ(「テーマ」を参照)を適用できます。このトピックでは、**C1DateTimePicker** コントロールのテーマを **C1ThemeRanierOrange** に変更します。

Blend の場合

次の手順に従います。

1. **[アセット]**パネルをクリックします。
2. 検索バーに、**「C1ThemeRainierOrange」**と入力します。
   [C1ThemeRainierOrange]アイコンが表示されます。
3. [C1ThemeRainierOrange]アイコンをダブルクリックしてプロジェクトに追加します。
4. 検索バーに「C1DateTimePicker」と入力して **C1DateTimePicker** コントロールを検索します。
5. **[C1DateTimePicker]**アイコンをダブルクリックして **C1DateTimePicker** コントロールをプロジェクトに追加します。
6. **[オブジェクトとタイムライン]**タブで**[C1DateTimePicker]**を選択し、これをドラッグアンドドロップ操作で**[C1ThemeRainierOrange]**の下に配置します。
7. プロジェクトを実行します。

Visual Studio の場合

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、**[C1ThemeRainierOrange]**アイコンをダブルクリックしてテーマを宣言します。このタグは次のようになります。

   XAML
   ```
   <my:C1ThemeRainierOrange></my:C1ThemeRainierOrange>
   ```
4. <my:C1ThemeRainierOrange> タグと </my:C1ThemeRainierOrange> タグの間にカーソルを置きます。
5. ツールボックスで、**[C1DateTimePicker]**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。そのタグは <my:C1ThemeRainierOrange> タグの子として表示され、マークアップは次のようになります。

   XAML
   ```
   <my:C1ThemeRainierOrange>
       <c1:C1DateTimePicker></c1:C1DateTimePicker>
   </my:C1ThemeRainierOrange>
   ```
6. プロジェクトを実行します。

このトピックの作業結果

次の図は、**C1ThemeRainierOrange** テーマが適用された C1DateTimePicker コントロールを示しています。

メモ:このトピックの内容は、ComponentOne Studio for Silverlight にのみ適用されます。

C1DatePicker コントロール

DatePicker for WPF/Silverlight を使用して、日付を選択して表示します。**C1DatePicker** コントロールは、日付の値を選択するための簡単なおよび直感的な UI を提供します。C1DatePicker のドロップダウン矢印を使用してカレンダーで日付を選択できます。

C1DatePicker クイックスタート

このクイックスタートは、**C1DatePicker** コントロールを初めて使用するユーザーのために用意されています。このクイックスタートでは、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 **C1DatePicker** コントロールを追加し、**C1DatePicker** コントロールをカスタマイズします。

手順 1：C1DatePicker コントロールを含むアプリケーションの作成

この手順では、WPF/Silverlight アプリケーションを作成し、ウィンドウに **C1DatePicker** コントロールを追加します。

次の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新規作成]**→**[プロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで**[WPF アプリケーション]**または**[Silverlight アプリケーション]**を選択します。
3. プロジェクトの**名前**と**場所**を入力し、**[OK]**をクリックして新しいアプリケーションを作成します。
4. ツールボックスで、**C1DatePicker** アイコンをダブルクリックして、**C1DatePicker** コントロールを WPF/Silverlight アプリケーションに追加します。

これで、**C1DatePicker for WPF/Silverlight** クイックスタートの最初の手順は終了です。この手順では、プロジェクトを作成し、それに **C1DatePicker** コントロールを追加しました。次の手順では、追加したコントロールをカスタマイズします。

手順 2：C1DatePicker のカスタマイズ

この手順では、**C1DatePicker** コントロールをカスタマイズします。

C1DatePicker コントロールを選択し、Visual Studio の**[プロパティ]**ウィンドウで次のプロパティを設定します。

- **SelectedDateFormat** プロパティを **Long** に設定します。これで、コントロールの日付書式が変わり、月と週の完全な名前が表示されます。
- **SelectedDate** プロパティの隣にあるドロップダウン矢印をクリックし、ドロップダウンカレンダーから **February 14**（または任意の日付）をクリックします。
- **ButtonBackground** プロパティの隣にあるドロップダウン矢印をクリックし、ドロップダウンカラーピッカーからカラーを選択します。

アプリケーションをカスタマイズできたので、プロジェクトを実行し、コントロールの実行時の動作を確認します。

手順 3：アプリケーションの実行

前の2つの手順では、**C1DatePicker** コントロールを含む WPF アプリケーションを作成し、コントロールをカスタマイズしました。このクイックスタートの最後の手順では、プロジェクトを実行し、コントロールを操作してみます。

[F5]キーを押してプロジェクトを実行します。ドロップダウン矢印が **ButtonBackground** プロパティで指定したカラーになって

います。**SelectedDate** が表示ボックスに表示され、月と週の完全な名前が表示されます。

おめでとうございます!

これでクイックスタートは終了です。クイックスタートが終了したら、次に「タスク別ヘルプ」を参照することをお勧めします。

C1DatePicker の使い方

以下のトピックでは、**C1DatePicker** コントロールの要素および機能について説明します。

C1DatePicker の要素

DateTimeEditors for WPF/Silverlight には、**C1DatePicker** コントロールがあります。このシンプルな日付ピッカーコントロールは、実行時にクリックして、ドロップダウンカレンダーから日付を選択するために使用できます。XAML ウィンドウに追加された **C1DatePicker** コントロールは、完全な機能を備えた日付ピッカーになります。デフォルトでは、コントロールのインターフェイスは次の図のように表示されます。

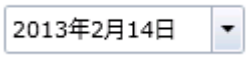
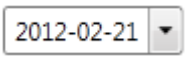
C1DatePicker コントロールは、次の要素で構成されます。

- **表示ボックス**
  表示ボックスには、選択されている日付が表示されます。この時刻は、**SelectedDate** プロパティを使用して設定できます。実行時に表示ボックスに直接数値を入力することもできます。入力した数値は、自動的に日付に変換されます。このコントロールは、**SelectedDate** プロパティを使用して、**Long**、**Short**(デフォルト)、**Custom** の3つの編集モードで日付を表示できます。
- **ドロップダウン矢印**
  実行時に **C1DatePicker** コントロールのドロップダウン矢印をクリックすると、表示されるドロップダウンカレンダーから日付を選択できます。

日付書式

SelectedDateFormat プロパティを使用すると、日付ピッカーを表示するときの書式を設定できます。**SelectedDateFormat** プロパティは、**Short**、**Long**、または **Custom** に設定できます。次の表に、それぞれの日付書式を示します。

日付書式	結果	説明

Short（デフォルト）		短い日付書式の数値で表示されます。
Long		長い日付書式が表示されます。
Custom		**CustomFormat** プロパティで定義されたカスタムの日付書式が表示されます。 **メモ**: 省略された書式でなく、完全な書式を使用してください。

レイアウトおよび外観

以下のトピックでは、**C1DatePicker** 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/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。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。

ComponentOne ClearStyle 技術

DateTimeEditors 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、コントロールのスタイルを簡単に設定できます。

次の表に、**C1DatePicker** コントロールのブラシのプロパティの概要を示します。

ブラシ	説明
Background	コントロールの背景のブラシを取得または設定します。
ButtonBackground	ボタンの背景色のブラシを取得または設定します。
ButtonForeground	ボタンの前景色のブラシを取得または設定します。
MouseOverBrush	マウスポインタが置かれたボタンを強調表示するために使用される System.Windows.Media.Brush を取得または設定します。
PressedBrush	クリックされたボタンを強調表示するために使用される System.Windows.Media.Brush を取得または設定します。

いくつかのプロパティを設定することで、**C1DateTimePicker** コントロールの外観を完全に変更できます。たとえば、**ButtonBackground** は、**C1DateTimePicker** コントロールのドロップダウン矢印の背景を設定します。**ButtonBackground** プロパティを "#FFC500FF" に設定すると、**C1DateTimePicker** コントロールは次のようになります。

外観のプロパティ

C1DatePicker コントロールには、コントロールの外観をカスタマイズするための複数のプロパティが含まれます。コントロールに表示されるテキストの外観を変更したり、コントロールのグラフィック要素をカスタマイズすることができます。以下のトピックでは、これらの外観プロパティの一部について説明します。

テキストのプロパティ

次のプロパティを使用して、**C1DatePicker** コントロールのテキストの外観をカスタマイズできます。

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

色のプロパティ

次のプロパティを使用して、コントロール自体に使用される色をカスタマイズできます。

プロパティ	説明
Background	コントロールの背景を描画するブラシを取得または設定します。これは依存プロパティです。
Foreground	前景色を描画するブラシを取得または設定します。これは依存プロパティです。

境界線のプロパティ

次のプロパティを使用して、コントロールの境界線をカスタマイズできます。

プロパティ	説明
BorderBrush	コントロールの境界線の背景を描画するブラシを取得または設定します。これは依存プロパティです。
BorderThickness	コントロールの境界線の太さを取得または設定します。これは依存プロパティです。

サイズのプロパティ

次のプロパティを使用して、**C1DatePicker** コントロールのサイズをカスタマイズできます。

プロパティ	説明
ActualHeight	この要素のレンダリングされた高さを取得または設定します。これは依存プロパティです。
ActualWidth	この要素のレンダリングされた幅を取得または設定します。これは依存プロパティです。
Height	要素の推奨高さを取得または設定します。これは依存プロパティです。
MaxHeight	要素の最大高さ制約を取得または設定します。これは依存プロパティです。
MaxWidth	要素の最大幅制約を取得または設定します。これは依存プロパティです。
MinHeight	要素の最小高さ制約を取得または設定します。これは依存プロパティです。
MinWidth	要素の最小幅制約を取得または設定します。これは依存プロパティです。
Width	要素の幅を取得または設定します。これは依存プロパティです。

テーマ

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。

C1DatePicker コントロールをプロジェクトに追加すると、次の図に示すように、コントロールはデフォルトの青色のテーマで表示されます。

C1DateTimePicker コントロールには、WPF の13のつのテーマ（BureauBlack、ExpressionDark、ExpressionLight、Office2007Black、Office2007Blue、Office2007Silver、Office2010Black、Office2010Blue、Office2010Silver、RainierOrange、ShinyBlue、WhistlerBlue）の中から1つテーマを設定できます。次の表に、テーマごとのサンプルを示します。

完全なテーマ名	外観
C1ThemeBureauBlack	
C1ThemeExpressionDark	
C1Blue (WPF のみ)	

C1ThemeExpressionLight	
C1ThemeOffice2007Black	
C1ThemeOffice2007Blue	
C1ThemeOffice2007Silver	

C1ThemeOffice2010Black	
C1ThemeOffice2010Blue	
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。

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、**C1DatePicker** 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。**C1DatePicker** コントロール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、**C1DatePicker** 製品を使っ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。

Null 値の許可

デフォルトでは、**C1DatePicker** コントロールは null 値の入力を許可しませんが、**AllowNull** プロパティを **True** に設定すると、コントロールに null 値の受け入れを強制できます。このトピックでは、デザイナ、XAML、およびコードで、**AllowNull** プロパティを **True** に設定する方法について説明します。

デザイナの場合

次の手順に従います。

1. **C1DatePicker** コントロールをクリックして選択します。
2. Visual Studio の**[プロパティ]**ウィンドウで、**AllowNull**チェックボックスをオンにします。

XAML の場合

null 値を許可するには、**AllowNull="True"** を <my:C1DatePicker>タグに配置します。マークアップは次のようになります。

XAML

```
<c1:C1DatePicker AllowNull="True"/>
```

コードの場合

次の手順に従います。

1. **MainWindow.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1DatePicker1.AllowNull = True
```

C#

```
c1DatePicker1.AllowNull = true;
```

3. プロジェクトを実行します。

日付書式の選択

デフォルトでは、**C1DatePicker** コントロールは短い書式で日付を表示しますが、日付を長い書式またはカスタム書式で表示することもできます。このトピックでは、デザイナ、XAML、およびコードで、日付書式を変更する方法について説明します。

デザイナの場合

日付書式を変更するには、次の手順に従います。

1. **C1DatePicker** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**SelectedDateFormat** のドロップダウン矢印をクリックし、リストからオプションを選択します。この例では、**[Long]**を選択します。

XAML の場合

日付書式を変更するには、**SelectedDateFormat="Long"** を <my:C1DatePicker> タグに配置します。マークアップは次のようになります。

XAML

```
<c1:C1DatePicker SelectedDateFormat="Long">
```

コードの場合

日付書式を変更するには、次の手順に従います。

1. **MainWindow.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

WPF

Visual Basic

```
C1DatePicker1.SelectedDateFormat =
C1.WPF.DateTimeEditors.C1DatePickerFormat.Long
```

C#

```
c1DateTimePicker1.SelectedDateFormat =
C1.WPF.DateTimeEditors.C1DatePickerFormat.Long;
```

Silverlight

Visual Basic

```
C1DatePicker1.SelectedDateFormat =
C1.Silverlight.DateTimeEditors.C1DatePickerFormat.Long
```

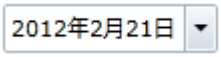
C#

```
C1DateTimePicker1.SelectedDateFormat =
C1.Silverlight.DateTimeEditors.C1DatePickerFormat.Long;
```

3. プロジェクトを実行します。

このトピックの作業結果

このトピックでは、**SelectedDateFormat** を **Long** に設定して、日付を長い書式で表示しました。最終的な結果は、次の画像のようになります。



週の最初の曜日の設定

デフォルトでは、**C1DatePicker** コントロールは、ドロップダウンカレンダーの週の最初の曜日を日曜日にしますが、この曜日は必要に応じて変更できます。このトピックでは、デザイナ、XAML、およびコードで、週の最初の曜日を変更する方法について説明します。

デザイナの場合

週の最初の曜日を変更するには、次の手順に従います。

1. **C1DatePicker** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**FirstDayOfWeek** のドロップダウン矢印をクリックし、リストから曜日を選択します。この例では、**[Monday]**を選択します。

XAML の場合

次のように <c1:C1DatePicker> タグのマークアップを変更します。

XAML

```
<c1:C1DatePicker FirstDayOfWeek="Monday">
```

コードの場合

日付書式を変更するには、次の手順に従います。

1. **MainWindow.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1DatePicker1.FirstDayOfWeek = DayOfWeek.Monday
```

C#

```
c1DatePicker1.FirstDayOfWeek = DayOfWeek.Monday;
```

3. プロジェクトを実行します。

このトピックの作業結果

このトピックでは、**FirstDayOfWeek** プロパティを **Monday** に設定して、ドロップダウンカレンダーで週の最初の曜日を月曜日にしました。最終的な結果は、次の画像のようになります。

カレンダーの開始日と終了日の設定

DisplayDateStart プロパティと **DisplayDateEnd** プロパティを設定して、ドロップダウンカレンダーに表示される日付を変更することができます。たとえば、カレンダーに3週間だけ表示する場合に、表示される最初の日付と最後の日付を設定することができます。このトピックでは、デザイナ、XAML、およびコードで、開始日と終了日を変更する方法について説明します。

デザイナの場合

カレンダーに表示される日付を変更するには、次の手順に従います。

1. **C1DatePicker** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**DisplayDateStart** のドロップダウン矢印をクリックし、リストから日付を選択します。この例では、**2/5/2012** を選択します。
3. **[プロパティ]**ウィンドウで、**DisplayDateEnd** のドロップダウン矢印をクリックし、リストから日付を選択します。この例では、**2/25/2012** を選択します。

XAML の場合

カレンダーの最初日と最後日を指定するには、マークアップが次のようになるように<c1:C1DatePicker>タグ内にDisplayDateStart = "02/05/2012"およびDisplayDateEnd = "02/25/2012"を配置します。

XAML

```
<c1:C1DatePicker DisplayDateEnd="02/25/2012" DisplayDateStart="02/05/2012">
```

コードの場合

カレンダーに表示される日付を変更するには、次の手順に従います。

1. MainWindow.xaml.cs ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
Dim dateStringStart As String = "02/05/2012"
Dim dateStringEnd As String = "02/25/2012"
C1DatePicker1.DisplayDateStart = DateTime.Parse(dateStringStart)
C1DatePicker1.DisplayDateEnd = DateTime.Parse(dateStringStart)
```

C#

```
string dateStringStart = "02/05/2012";
string dateStringEnd = "02/25/2012";
C1DatePicker1.DisplayDateStart = DateTime.Parse(dateStringStart);
C1DatePicker1.DisplayDateEnd = DateTime.Parse(dateStringEnd);
```

3. プロジェクトを実行します

このトピックの作業結果

このトピックでは、**DisplayDateStart** プロパティと **DisplayDateEnd** プロパティを設定して、ドロップダウンカレンダーに表示される日付を決定しました。最終的な結果は、次の画像のようになります。

テーマを使用する (Silverlightのみ)

C1DatePicker コントロールでは、複数のテーマをコントロールに適用することができます。このトピックでは、**C1DatePicker** コントロールのテーマを **C1ThemeRanierOrange** に変更します。使用可能なテーマの詳細については、「テーマ」を参照してください。

Blend の場合

次の手順に従います。

1. **[アセット]**タブをクリックします。
2. 検索バーに、「C1ThemeRainierOrange」と入力します。**[C1ThemeRainierOrange]**アイコンが表示されます。
3. **[C1ThemeRainierOrange]**アイコンをダブルクリックしてプロジェクトに追加します。
4. 検索バーに「C1DatePicker」と入力して **C1DatePicker** コントロールを検索します。
5. **C1DatePicker**アイコンをダブルクリックして **C1DatePicker** コントロールをプロジェクトに追加します。
6. **[オブジェクトとタイムライン]**タブで**[C1DatePicker]**を選択し、これをドラッグ &ドロップ操作で**[C1ThemeRainierOrange]**の下に配置します。
7. プロジェクトを実行します。

Visual Studio の場合

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、**[C1ThemeRainierOrange]**アイコンをダブルクリックしてテーマを宣言します。このタグは次のようになります。<c1:C1ThemeRainierOrange></c1:C1ThemeRainierOrange>
4. <c1:C1ThemeRainierOrange> タグと </c1:C1ThemeRainierOrange> タグの間にカーソルを置きます。
5. ツールボックスで、**[C1DatePicker]**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。そのタグは <c1:C1ThemeRainierOrange > タグの子として表示され、マークアップは次のようになります。

```
XAML
<c1:C1ThemeRainierOrange>
    <c1:C1DatePicker/>
</c1:C1ThemeRainierOrange>
```

6. プロジェクトを実行します。

メモ:このトピックの内容は、ComponentOne Studio for Silverlight にのみ適用されます。

C1TimeEditor コントロール

TimePicker for WPF/Silverlight を使用して、エンドユーザーと時刻情報を交換します。このコントロールは、時刻の値を選択するための簡単なインターフェイスを提供します。時刻は、スピンボタンまたはキーボードの矢印キーを使用するか、フィールドに値を入力して選択できます。

クイックスタート

このクイックスタートは、**C1TimeEditor** コントロールを初めて使用するユーザーのために用意されています。このクイックスタートでは、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 **C1TimeEditor** コントロールを追加し、**C1TimeEditor** コントロールをカスタマイズします。

手順 1：C1TimeEditor コントロールを含むアプリケーションの作成

この手順では、最初に Visual Studio で **C1TimeEditor** コントロールを使用する WPF/Silverlight アプリケーションを作成します。

WPF プロジェクトを作成する場合、以下の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新しいプロジェクト]**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**[WPF アプリケーション]**を選択します。
3. プロジェクトの**名前**と**場所**を入力し、**[OK]**をクリックして新しいアプリケーションを作成します。
4. ツールボックスで、**C1TimeEditor** アイコンをダブルクリックして、**C1TimeEditor** コントロールを WPF アプリケーションに追加します。

Silverlight プロジェクトを作成する場合、以下の手順に従います。

1. Expression Blend で、[**ファイル**]→[**新しいプロジェクト**]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、左ペインからプロジェクトの種類として[Silverlight]を選択し、右ペインから**[Silverlight アプリケーション + Web サイト]**を選択します。
3. プロジェクトの**名前**と**場所**を入力し、**[OK]**をクリックします。Blend によって作成された新しいアプリケーションが開き、デザインビューに **MainPage.xaml** ファイルが表示されます。
4. の手順に従って、**C1TimeEditor** コントロールをプロジェクトに追加します。
    a. メニューから**[ウィンドウ]**→**[アセット]を選択して[アセット]**パネルを開きます。
    b. **[アセット]**パネルで、検索バーに「C1TimeEditor」と入力します。
    c. **C1TimeEditor** コントロールのアイコンが表示されます。
    d. **[C1TimeEditor]**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ます。

この手順では、プロジェクトを作成し、それに **C1TimeEditor** コントロールを追加しました。次の手順では、追加したコントロールをカスタマイズします。

手順 2：コントロールのカスタマイズ

この手順では、Blend とコードの両方を使って **C1TimeEditor** コントロールをカスタマイズします。

WPF のプロジェクトで C1TimeEditor コントロールを選択し、[プロパティ]ウィンドウで次のプロパティを設定します。

- **Format** プロパティを **ShortTime** に設定します。これにより、コントロールの時刻書式が変更され、時間と分のみが表示されるようになります。
- **Increment** プロパティを「01:00:00」に設定します。これにより、ユーザーがスピンボタンをクリックするたびに、コントロールの値が1時間単位で変化するようになります。
- **Interval** プロパティを「1000」に設定します。こうすると、1秒間待ってからコントロールの値が変化するようになります。
- **Value** プロパティを「17:00:00」に設定します。

Silverlight のプロジェクトでは、Blend で C1TimeEditor コントロールを選択し、[プロパティ]パネルで次のプロパティを設定します。

1. **Format** プロパティを **ShortTime** に設定します。これにより、コントロールの時刻書式が変更され、時間と分のみが表示されるようになります。
2. **Increment** プロパティを "01:00:00" に設定します。これにより、ユーザーがスピンボタンをクリックするたびに、コントロールの値 が1時間単位で変化するようになります。
3. **Interval** プロパティを "1000" に設定します。こうすると、1秒間待ってからコントロールの値が変化するようになります。
4. 次の手順に従って、現在の時刻を午後5時に設定します。
    a. XAML エディタで、x:Name="C1TimeEditor1" を タグに追加します。これで、このコントロールをコードから呼び出すための一意の識別子が指定されます。
    b. **MainPage.xaml.cs** または **MainPage.xaml.vb** ページを開きます。
    c. **InitializeComponent()** メソッドの下に次のコードを追加します。

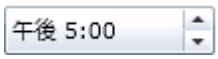
アプリケーションをカスタマイズできたので、プロジェクトを実行し、コントロールの実行時の動作を確認します。

手順 3：アプリケーションの実行

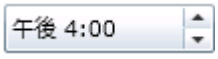
前の2つの手順では、**C1TimeEditor** コントロールを含む WPF/Silverlight アプリケーションを作成し、コントロールをカスタマイズしました。このクイックスタートの最後の手順では、プロジェクトを実行し、コントロールを操作してみます。

次の手順に従います。

1. **[F5]**キーを押してプロジェクトを実行します。時刻の値が「午後 5:00」の状態でコントロールがロードされ、コントロールに時間と分のみが表示されていることを確認します。



2. 時刻を戻すボタン をクリックして、時刻の値が1時間戻って午後4時になることを確認します。



3. 時刻を進めるボタン をクリックして押したままにし、コントロールの値が増加し続けるようにします。値が変化するたびに、コントロールが1秒間待機することを確認します。

おめでとうございます!

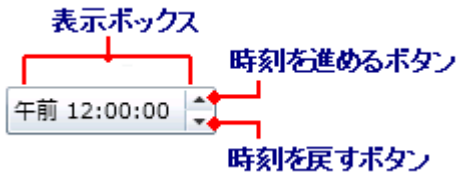
これでクイックスタートは終了です。このクイックスタートは終了したので、次に「C1TimeEditor for WPF の使い方」または「タスク別ヘルプ」のトピックを参照することをお勧めします。

C1TimeEditor for WPF の使い方

次のトピックでは、**C1TimeEditor** コントロールのすべての要素といくつかの機能について説明します。

C1TimeEditor の要素

DateTimeEditors for WPF/Silverlight には **C1TimeEditor** コントロールがあります。これは、短い書式の時刻、長い書式の時刻、および時間間隔を表示できる簡単な時刻ピッカーを提供します。XAML ウィンドウに追加された **C1TimeEditor** コントロールは、完全な機能を備えた時刻ピッカーになります。デフォルトでは、コントロールのインターフェイスは次の図のように表示されます。



C1TimeEditor コントロールは、次の要素で構成されます。

- **表示ボックス**
  表示ボックスには、選択されている時刻が表示されます。この時刻は、**Value** プロパティを使って設定できます。表示ボックスに直接数値を入力することもできます。入力した数値は、自動的に時刻に変換されます。このコントロールは、**LongTime**（デフォルト）、**ShortTime**、**TimeSpan** の3つの編集モードで時刻を表示できます。
- **時刻を進めるボタン**
  時刻を進めるボタンを使用すると、時刻ピッカーに表示する時刻を進めることができます。進めるボタンをクリックすると、別に指定されていない限り、時刻が1分進みます。
- **時刻を戻すボタン( )**
  時刻を戻すボタン( ) を使用すると、時刻ピッカーに表示する時刻を戻すことができます。戻すボタンをクリックすると、別に指定されていない限り、時刻が1分戻ります。

スピン間隔

スピンボタンを使って値を増減させる方法には、2通りあります。どちらかのボタンを繰り返しクリックすると、時刻を任意のペースで進めたり戻すことができます。また、時刻を戻すボタン( )または時刻を進めるボタンを押したままにすると、プログラムで指定されている間隔で時刻を進めたり戻すことができます。この間隔を指定するには、**Interval** プロパティを設定します。

デフォルトでは、**Interval** プロパティは 33 ミリ秒に設定されており、時刻の値を高速にスクロールできます。このスクロール速度を遅くするには、500 ミリ秒（2分の1秒）のように大きい数値を指定します。速くするには、10 ミリ秒（100 分の1秒）のように小さい数値を指定します。**Interval** を "0" に設定することはできません。

値のインクリメント値

ユーザーが時刻を進める/戻すスピンボタンをクリックするたびに、コントロールの値は、プログラムで指定されているインクリメント値だけ増加または減少します。デフォルトでは、このインクリメント値は 00:01:00（1分）です。このインクリメント値を大きくまたは小さくするには、**Increment** プロパティを設定します。**Increment** プロパティは、00:00:00（スピンボタンが無効になります）から 23:59:59 までの任意の値に設定できます。

時刻書式

Format プロパティを使用すると、日付ピッカーに表示する書式を設定できます。Format プロパティには、**ShortTime**、**LongTime**、または **TimeSpan** を 設定できます。次の表に、それぞれの時刻書式を示します。

時刻書式	結果	説明
ShortTime	12:00 AM	秒を含まない短い書式で時刻が表示されます。
LongTime (default)	12:00:00 AM	秒を含む長い書式で時刻が表示されます。
TimeSpan	00:00:00	時間間隔が表示され、a.m./p.m. の別は表示されません。

レイアウトおよび外観

以下のトピックでは、**C1TimeEditor** 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/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。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。

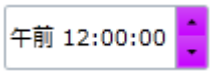
ComponentOne ClearStyle 技術

TimeEditor 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、グリッド全体のスタイルを簡単に設定できます。

次の表に、**C1TimeEditor** コントロールのブラシのプロパティの概要を示します。

ClearStyle プロパティ	説明
Background	コントロールの背景のブラシを取得または設定します。
ButtonBackground	ボタンの背景色のブラシを取得または設定します。
ButtonForeground	ボタンの前景色のブラシを取得または設定します。
MouseOverBrush	マウスポインタが置かれたボタンを強調表示するために使用される System.Windows.Media.Brush を取得または設定します。
PressedBrush	クリックされたボタンを強調表示するために使用される System.Windows.Media.Brush を取得または設定します。

いくつかのプロパティを設定することで、**C1TimeEditor** コントロールの外観を完全に変更できます。たとえば、**ButtonBackground** は、**C1TimeEditor** コントロールのドロップダウンボタンの背景色を設定します。**ButtonBackground** プロパティを "#FFC500FF" に設定すると、**C1TimeEditor** コントロールは次のようになります。



外観のプロパティ

C1TimeEditor コントロールには、コントロールの外観をカスタマイズするための複数のプロパティが含まれます。コントロールに表示されるテキストの外観を変更したり、コントロールのグラフィック要素をカスタマイズすることができます。以下のトピックでは、これらの外観プロパティの一部について説明します。

テキストのプロパティ

次のプロパティを使用して、**C1TimeEditor** コントロールのテキストの外観をカスタマイズできます。

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

色のプロパティ

次のプロパティを使用して、コントロール自体に使用される色をカスタマイズできます。

プロパティ	説明
Background	コントロールの背景を描画するブラシを取得または設定します。これは依存プロパティです。
Foreground	前景色を描画するブラシを取得または設定します。これは依存プロパティです。

境界線のプロパティ

次のプロパティを使用して、コントロールの境界線をカスタマイズできます。

プロパティ	説明
BorderBrush	コントロールの境界線の背景を描画するブラシを取得または設定します。これは依存プロパティです。
BorderThickness	コントロールの境界線の太さを取得または設定します。これは依存プロパティです。

サイズのプロパティ

次のプロパティを使用して、**C1TimeEditor** コントロールのサイズをカスタマイズできます。

プロパティ	説明
ActualHeight	この要素のレンダリングされた高さを取得または設定します。これは依存プロパティです。
ActualWidth	この要素のレンダリングされた幅を取得または設定します。これは依存プロパティです。
Height	要素の推奨高さを取得または設定します。これは依存プロパティです。
MaxHeight	要素の最大高さ制約を取得または設定します。これは依存プロパティです。
MaxWidth	要素の最大幅制約を取得または設定します。これは依存プロパティです。
MinHeight	要素の最小高さ制約を取得または設定します。これは依存プロパティです。
MinWidth	要素の最小幅制約を取得または設定します。これは依存プロパティです。
Width	要素の幅を取得または設定します。これは依存プロパティです。

テーマ

DateTimeEditors for WPF/Silverlight には、コントロールの外観をカスタマイズするためのテーマがいくつか組み込まれています。ページに DateTimeEditors for WPF/Silverlight コントロールのいずれかを初めて追加すると、次の画像のようになります。

C1DateTimePicker コントロールには、WPF の13のつのテーマ（BureauBlack、ExpressionDark、ExpressionLight、Office2007Black、Office2007Blue、Office2007Silver、Office2010Black、Office2010Blue、Office2010Silver、RainierOrange、ShinyBlue、WhistlerBlue）の中から1つテーマを設定できます。次の表に、テーマごとのサンプルを示します。

テーマ名	テーマのプレビュー
C1ThemeBureauBlack	
C1ThemeExpressionDark	
C1Blue (WPF のみ)	
C1ThemeExpressionLight	
C1ThemeOffice2007Black	
C1ThemeOffice2007Blue	
C1ThemeOffice2007Silver	
C1ThemeOffice2010Black	
C1ThemeOffice2010Blue	
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。

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、**C1TimeEditor** 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。**C1TimeEditor** コントロール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、**C1TimeEditor** コントロールを使っ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。

null 値の許可

デフォルトでは、**C1TimeEditor** コントロールは null 値の入力を許可しませんが、**AllowNull** プロパティを **True** に設定すると、コントロールに null 値の受け入れを強制できます。このトピックでは、デザイナ、XAML、およびコードで、**AllowNull** プロパティを **True** に設定する方法について説明します。

デザイナの場合

次の手順に従います。

1. **C1TimeEditor** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**[AllowNull]**チェックボックスをオンにします。

XAML の場合

null 値を許可するには、AllowNull="True" を <my:C1TimeEditor>タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1TimeEditor AllowNull="True"/>
```

コードの場合

次の手順に従います。

1. **MainWindow.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1TimeEditor1.AllowNull = True
```

C#

```
c1TimeEditor1.AllowNull = true;
```

3. プロジェクトを実行します。

スピンボタンの削除

ShowButtons プロパティを **False** に設定して、**C1TimeEditor** コントロールのスピンボタンを削除することができます。このトピックでは、デザイナ、XAML、およびコードで、**ShowButtons** プロパティを **False** に設定する方法について説明します。

デザイナの場合

次の手順に従います。

1. **C1TimeEditor** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**[ShowButtons]**チェックボックスをオフにします。

XAML の場合

スピンボタンを削除するには、ShowButtons="False" を タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1TimeEditor ShowButtons="False"/>
```

コードの場合

次の手順に従います。

1. **Window1.xaml.**cs ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します

VisualBasic

```
C1TimeEditor1.ShowButtons = False
```

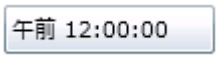
C#

```
c1TimeEditor1.ShowButtons = false;
```

3. プロジェクトを実行します。

このトピックの作業結果

スピンボタンを削除した **C1TimeEditor** コントロールは、次の図のように表示されます。



時刻書式の選択

デフォルトでは、**C1TimeEditor** コントロールには、秒を含む長い書式で時刻が表示されますが、短い書式や時間間隔書式で時刻を表示することもできます。このトピックでは、デザイナ、XAML、およびコードで、時刻書式を変更する方法について説明します。

デザイナの場合

時刻書式を変更するには、次の手順に従います。

1. **C1TimeEditor** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**Format** のドロップダウン矢印をクリックし、リストからモードを選択します。この例では、**[ShortTime]**を選択します。

XAML の場合

時刻書式を変更するには、**Format="ShortTime"** を タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1TimeEditor Format="ShortTime">
```

コードの場合

時刻書式を変更するには、次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. 次の名前空間をインポートします。

WPF

Visual Basic

```
Imports C1.WPF.DateTimeEditors
```

C#

```
using C1.WPF.DateTimeEditors;
```

Silverlight

Visual Basic

```
Imports C1.Silverlight.DateTimeEditors
```

C#

```
using C1.Silverlight.DateTimeEditors;
```

3. **InitializeComponent()** メソッドの下に次のコードを追加します。

WPF

Visual Basic

```
C1TimeEditor1.Format = C1TimeEditorFormat.ShortTime
```

C#

```
c1TimeEditor1.Format = C1TimeEditorFormat.ShortTime;
```

Silverlight

Visual Basic

```
C1TimeEditor1.TimeFormat = C1TimeEditorFormat.ShortTime
```

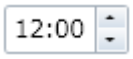
C#

```
C1TimeEditor1.TimeFormat = C1TimeEditorFormat.ShortTime;
```

4. プロジェクトを実行します。

このトピックの作業結果

このトピックでは、Format を **ShortTime** に設定して、短い書式で時刻を表示しました。最終的な結果は、次の画像のようになります。



スピン間隔の設定

デフォルトでは、**Interval** プロパティは 33 ミリ秒に設定されており、時刻の値を高速にスクロールできます。このトピックでは、**Interval** プロパティを 1000 ミリ秒に設定して、値が変化する時間間隔を長く指定します。スピン間隔の詳細については、「スピン間隔」を参照してください。

デザイナの場合

次の手順に従います。

1. C1TimeEditor コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**Interval** プロパティを見つけ、そのテキストボックスに「1000」と入力します。
3. プロジェクトを実行し、時刻を進めるボタン()をクリックして押したままにします。値が1秒に1回だけ増加することを確認します。

XAML の場合

次の手順に従います。

1. **Interval="1000"** を <my:C1TimeEditor>タグに追加します。マークアップは次のようになります。

```
XAML
<my:C1TimeEditor Interval="1000"/>
```

2. プロジェクトを実行し、時刻を進めるボタン()をクリックして押したままにします。値が1秒に1回だけ増加することを確認します。

コードの場合

次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1TimeEditor1.Interval = 1000
```

C#

```
c1TimeEditor1.Interval = 1000;
```

3. プロジェクトを実行し、時刻を進めるボタン()をクリックして押したままにします。値が1秒に1回だけ増加することを確認します。

インクリメント値の設定

デフォルトでは、**C1TimeEditor** コントロールの時刻は、1分単位で増減するように設定されています。このインクリメント値は、Increment プロパティを任意の時間インクリメント値に設定して変更できます。このトピックでは、**C1TimeEditor** コントロールの時間インクリメント値を1時間 30 分に設定します。時間インクリメント値の詳細については、「値のインクリメント値」を参照してください。

デザイナの場合

次の手順に従います。

1. **C1TimeEditor** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**Increment** プロパティを見つけ、そのテキストボックスに**「01:30:00」**と入力します。
3. プロジェクトを実行し、時刻を進めるボタン()をクリックします。時刻が1時間 30 分先に進むことを確認します。

XAML の場合

次の手順に従います。

1. **Increment="01:30:00"** を <my:C1TimeEditor>タグに追加します。マークアップは次のようになります。

```
XAML
<my:C1TimeEditor Increment="01:30:00"/>
```

2. プロジェクトを実行し、時刻を進めるボタン()をクリックします。時刻が1時間 30 分先に進むことを確認します。

コードの場合

次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1TimeEditor1.Increment = New TimeSpan(01, 30, 00)
```

C#

```
c1TimeEditor1.Increment = new TimeSpan(01,30,00);
```

3. プロジェクトを実行し、時刻を進めるボタン()をクリックします。時刻が1時間 30 分先に進むことを確認します。

現在時刻の指定

C1TimeEditor コントロールの現在の時刻を指定するには、デザイナ、XAML、またはコードを使って **Value** プロパティを設定します。

メモ:Value プロパティを XAML で文字列として設定しないでください。この値を文字列から解析する操作は、カルチャに依存します。現在のカルチャで値を設定したが、ユーザーが別のカルチャを使用している場合は、サイトを読み込む際に XamlParseException を受け取る可能性があります。最善の方法は、これらの値をコードまたはデータ連結を介して設定することです。

デザイナの場合

次の手順に従います。

1. **C1TimeEditor** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、**Value** プロパティを**「07:00:00」**に設定します。コントロールの時刻は午前 7:00:00 になります。

XAML の場合

現在の時刻を指定するには、Value="07:00:00" を <my:C1TimeEditor> タグに配置します。マークアップは次のようになります。

XAML

```
<my:C1TimeEditor Value="07:00:00"/>
```

コントロールの時刻は午前 7:00:00 になります。

コードの場合

次の手順に従います。

1. **Window1.xaml.cs** ページを開きます。
2. **InitializeComponent()** メソッドの下に次のコードを追加します。

VisualBasic

```
C1TimeEditor1.Value = New TimeSpan(7, 0, 0)
```

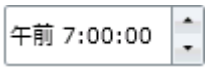
C#

```
c1TimeEditor1.Value = new TimeSpan(7,0,0);
```

3. プロジェクトを実行し、コントロールの時刻が午前 7:00:00 になることを確認します。

このトピックの作業結果

このトピックで示されている手順に従って、**C1TimeEditor** コントロールの時刻を午前 7:00:00 に変更しました。結果は次のようになります。



時間間隔の操作

C1TimeEditor コントロールに時間間隔が表示されるように変更できます。このチュートリアルでは、5:00 から 10:00 までの時

間間隔を表す **C1TimeEditor** コントロールを作成します。また開始値を午前 7:00 に設定するプロジェクトのコードも記述します。

次の手順に従います。

1. **C1TimeEditor** コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウで、次の手順に従います。
    - Format プロパティを **TimeSpan** に設定します。
    - **Maximum** プロパティに値(「10:00:00」など)を設定します。
    - **Minimum** プロパティに値(「05:00:00」など)を設定します。
    - **Value** プロパティに値(「07:00:00」など)を設定します。
3. プロジェクトを実行し、コントロールの時刻が 午前 07:00:00 の状態でロードされることを確認します。
4. 時刻を進めるボタン( )を、それ以上進めることができなくなるまでクリックします。コントロールの表示は、10:00:00 で止まります。
5. 時刻を戻すボタン( )を、それ以上戻すことができなくなるまでクリックします。コントロールの表示は、05:00:00 で止まります。

テーマを使用する (Silverlightのみ)

C1TimeEditor コントロールには、デフォルトで明るい青色のテーマが設定されていますが、全部で 複数のテーマ(「テーマ」を参照)を適用できます。このトピックでは、**C1TimeEditor** コントロールのテーマを **C1ThemeRanierOrange** に変更します。

Blend の場合

次の手順に従います。

1. **[アセット]**パネルをクリックします。
2. 検索バーに、「C1ThemeRainierOrange」と入力します。**C1ThemeRainierOrange**アイコンが表示されます。
3. **[C1ThemeRainierOrange]**アイコンをダブルクリックしてプロジェクトに追加します。
4. 検索バーに「C1TimeEditor」と入力して **C1TimeEditor** コントロールを検索します。
5. **[C1TimeEditor]**アイコンをダブルクリックして **C1TimeEditor** コントロールをプロジェクトに追加します。
6. **[オブジェクトとタイムライン]**タブで**[C1TimeEditor]**を選択し、これをドラッグアンドドロップ操作で**[C1ThemeRainierOrange]**の下に配置します。
7. プロジェクトを実行します。

Visual Studio の場合

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、**[C1ThemeRainierOrange]**アイコンをダブルクリックしてテーマを宣言します。このタグは次のようになります。

    <my:C1ThemeRainierOrange> </my:C1ThemeRainierOrange>

4. <my:C1ThemeRainierOrange> タグと </my:C1ThemeRainierOrange> タグの間にカーソルを置きます。
5. ツールボックスで、[C1TimeEditor]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。そのタグは <my:C1ThemeRainierOrange> タグの子として表示され、マークアップは次のようになります。

XAML

```
<my:C1ThemeRainierOrange>
    <c1:C1TimeEditor></c1:C1TimeEditor>
</my:C1ThemeRainierOrange>
```

6. プロジェクトを実行します。

このトピックの作業結果

次の図は、**C1ThemeRainierOrange** テーマが適用された **C1TimeEditor** コントロールを示しています。

メモ:このトピックの内容は、ComponentOne Studio for Silverlight にのみ適用されます。