DateTimeEditors for UWP

2018.03.06 更新

グレープシティ株式会社

DateTimeEditors for UWP

目次

DateTimeEditors for UWP

6つの基本的な日付/時刻エディタを使用できます。クラシックなデザインから Modern UI スタイルのデザインまで揃っています。**C1DateSelector**、**C1TimeSelector**、**C1DateTimeSelector** の各コントロールは、**DateTime** 入力を収集するための一連のドロップダウンを提供します。**C1DatePicker** コントロールは、最も使い慣れた形式のドロップダウンカレンダーを提供します。**C1DateTimePicker** コントロールは、**C1DatePicker** と **C1TimeEditor** を組み合わせた**1**つの大きなコントロールです。

主な特長

DateTimeEditors for UWP を使用すると、機能豊富でカスタマイズされたアプリケーションを作成できます。**DateTimeEditors for UWP** は、次の主な特長を備えています。

- **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）ボタンを使用して日付/時刻を選択できます。
- **null 値のサポート**
  デフォルトでは、C1DateTimePicker、**C1DatePicker**、**C1TimeEditor** の各コントロールで null 値を入力できます。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。

DateTimeSelector for UWP

DateTimeSelector for UWP を使用して、日付/時刻の情報を交換します。この機能は、日付/時刻の値、日付のみの値、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。日付/時刻は、ボタンを使用するか、キーボードの矢印を使用して選択できます。

クイックスタート

このクイックスタートガイドは、**DateTimeSelector for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートガイドでは、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 C1DateTimeSelector コントロールを追加し、C1DateTimeSelector コントロールをカスタマイズします。

手順1：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 C1DateTimeSelector for UWP を使用する UWP スタイルの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**]→[**新規作成**]→[**プロジェクト**]を選択します。
2. [**新しいプロジェクト**]ダイアログボックスで、左ペインの言語を展開し、言語の下で[**Windows ストア**]を選択し、テンプレートリストで[**新しいアプリケーション (XAML)**]を選択します。名前を入力し、[**OK**]をクリックしてプロジェクトを作成します。
3. MainPage.xaml が開いていない場合は開きます。`<Grid>` タグと `<Grid>` タグの間にカーソルを置き、1回クリックします。
4. ツールボックスに移動し、C1DateTimeSelector 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。

マークアップ

```
<Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}"
x:Name="LayoutRoot">
    <Xaml:C1DateTimeSelector HorizontalAlignment="Left"
VerticalAlignment="Top"/>
</Grid>
```

これで、**C1DateTimePicker** コントロールを含む UWP スタイルのアプリケーションを作成できました。次の手順では、コントロールの外観を変更します。

手順2：コントロールのカスタマイズ

前の手順では、C1DateTimeSelector コントロールを使用して UWP スタイルの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。

次の手順に従います。

1. アプリケーションの[プロパティ]ウィンドウで、**Brush** カテゴリをクリックしてオプションを表示します。**Background**、**BorderBrush**、**Foreground** の各プロパティのカスタム色を設定できます。
   以下の XAML マークアップを タグに挿入すると、C1DateTimeSelector コントロールの外観も変更できます。

マークアップ

```
Background="#CC66709C" Foreground="White"
```

2. コントロールの Text プロパティをカスタマイズして、**FontFamily** を Plantagenet Cherokee に変更し、**FontSize** を 20

ピクセルに設定します。
以下の XAML マークアップを <Xaml:C1DateTimeSelector/> タグに挿入すると、**FontFamily** および **FontSize** も変更できます。

マークアップ
FontFamily="Plantagenet Cherokee" FontSize="20"

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：プロジェクトの実行

前の手順では、C1DateTimeSelector 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。C1DateTimeSelector コントロールは次の図のようになります。

おめでとうございます。これで **C1DateTimeSelector for UWP** クイックスタートは終了です。

C1DateTimeSelector の使い方

以下のトピックでは、**C1DateTimePicker** コントロールの要素および機能について説明します。

C1DateTimeSelector の要素

DateTimeEditors for UWP には、C1DateTimeSelector コントロールが含まれています。これは、デフォルトで日付セレクタと時刻セレクタ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 C1DateTimeSelector コントロールは、完全な機能を備えた日付/時刻セレクタになります。デフォルトでは、コントロールのインタフェースは次の図のように表示されます。

C1DateTimeSelector コントロールは、次の要素で構成されます。

1. **日付セレクタ**
   日付セレクタは、月フィールド、日付と曜日のセレクタ、および年セレクタの3つのフィールドで構成されます。月、日付と曜日、および年を設定するには、ドロップダウン矢印を使用し、ドロップダウンリストから月、日付と曜日、および年を選択します。
2. **時刻セレクタ**
   時刻セレクタは、時間フィールド、分フィールド、および AM/PM フィールドの3つのフィールドで構成されます。時間、分、および AM/PM を設定するには、ドロップダウン矢印を使用し、ドロップダウンリストから時間、分、および AM/PM の設定を選択します。

C1DateTimeSelector の編集モード

デフォルトでは、ページに配置された C1DateTimeSelector コントロールには日付セレクタと時刻セレクタの両方が表示されます。表示されるセレクタは、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。日付セレクタのみを表示する場合は EditMode プロパティを **Date** に、時刻セレクタのみを表示する場合は EditMode プロパティを **Time** に、日付セレクタと時刻セレクタ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	

C1DateTimeSelector の外観プロパティ

C1DateTimeSelector 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。次の表では、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、C1DateTimeSelector 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、C1DateTimeSelector 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

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。
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。

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、C1DateTimeSelector 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。C1DateTimeSelector コントロールを初めて使用される場合は、まず「**C1DateTimeSelector 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、C1DateTimeSelector 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。

コントロールをコード内で作成する

このトピックでは、C1DateTimeSelector コントロールを作成し、コードを使用してコントロールに基本スタイルを適用する手順について説明します。

次の手順に従います。

1. MainPage.xaml が開いていない場合は開き、`<Grid> </Grid>` のタグセットを探します。次のようにマークアップを編集します。

マークアップ

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}" x:Name="LayoutRoot">
</Grid>
```

2. ページを右クリックし、リストから[**コードの表示**]を選択してコードビューに切り替えます。
3. 次の名前空間をアプリケーションに追加します。

C#

```
using C1.Xaml;
```

Visual Basic

```
Imports C1.Xaml
```

4. **InitializeComponent()** メソッドのすぐ下に、次のコンストラクタを挿入します。

Visual Basic

```
Dim c1DTS1 As C1DateTimeSelector = New C1DateTimeSelector
```

C#

```
C1DateTimeSelector c1DTS1 = new C1DateTimeSelector();
```

5. コンストラクタ内に次のコードを追加して、C1DateSelector の幅と高さを制御します。

Visual Basic

```
c1DTS1.Height = 55
c1DTS1.Width = 600
```

C#

```
c1DTS1.Height = 55;
```

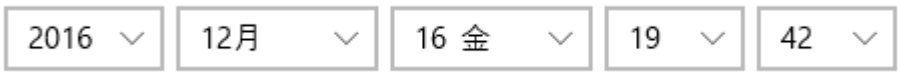
```
c1DTS1.Width = 600;
```

6. C1DateSelector を表示する行をコードの最後に挿入します。

Visual Basic
```
LayoutRoot.Children.Add(c1DTS1)
```

C#
```
LayoutRoot.Children.Add(c1DTS1);
```

7. アプリケーションを実行します。C1DateTimeSelector コントロールは次の図のようになります。



編集モードを選択する

デフォルトでは、C1DateTimeSelector コントロールには日付/時刻セレクタの両方が表示されますが、日付セレクタまたは時刻セレクタのみを表示することもできます。このトピックでは、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。

XAML の場合

編集モードを選択するには、次のいずれかのサンプルのように XAML マークアップを編集します。

XAML マークアップ
```
<Xaml:C1DateTimeSelector EditMode="DateTime"/>
```

XAML マークアップ
```
<Xaml:C1DateTimeSelector EditMode="Date"/>
```

XAML マークアップ
```
<Xaml:C1DateTimeSelector EditMode="Time"/>
```

コードの場合

編集モードを選択するには、次のコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic
```
c1DTS1.EditMode = C1.Xaml.C1DateTimePickerEditMode.Date
```

▶ C# コードの書き方

C#
```
c1DTS1.EditMode = C1.Xaml.C1DateTimePickerEditMode.Date;
```

選択されている日付を指定する

C1DateTimeSelector コントロールの時刻と日付を指定するには、XAML またはコードで SelectedDate プロパティを設定します。

XAML の場合

最初に選択されている日付を指定するには、XAML マークアップを次のように編集します。

XAML マークアップ
<Xaml:C1DateTimeSelector SelectedDate="1932, 10, 12/>

コードの場合

▶ Visual Basic コードの書き方

Visual Basic
c1DTS1.SelectedDate = New DateTime(1932, 10, 12)

▶ C# コードの書き方

C#
c1DTS1.SelectedDate = new DateTime(1932, 10, 12);

カレンダーの最小日と最大日を設定する

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、ユーザーが選択できる日付の範囲を変更できます。

アプリケーションに次のコードを追加して、カレンダーの最小日と最大日を設定します。

▶ Visual Basic コードの書き方

Visual Basic
c1DTS1.MinDate = New DateTime(1915, 6, 15) c1DTS1.MaxDate = New DateTime(2015, 6, 15)

▶ C# コードの書き方

C#
c1DTS1.MinDate = new DateTime(1915, 06, 15); c1DTS1.MaxDate = new DateTime(2015, 06, 15);

DateSelector for UWP

DateSelector for UWP を使用して、日付情報を表示および選択できます。C1DateSelector コントロールは、日付値を選択するための簡単で直感的な UI を提供します。C1DateSelector のドロップダウン矢印をクリックし、月、日付と曜日、および年を選択します。

クイックスタート

このクイックスタートガイドは、**DateSelector for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートガイドでは、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 C1DateSelector コントロールを追加し、C1DateSelector コントロールをカスタマイズします。

手順1：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 **C1DateSelector for UWP** を使用する UWP スタイルの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**]→[**新規作成**]→[**プロジェクト**]を選択します。
2. [**新しいプロジェクト**]ダイアログボックスで、左ペインの言語を展開し、言語の下で[**Windows ストア**]を選択し、テンプレートリストで[**新しいアプリケーション (XAML)**]を選択します。**名前**を入力し、[**OK**]をクリックしてプロジェクトを作成します。
3. MainPage.xaml が開いていない場合は開きます。`<Grid>` タグと `<Grid>` タグの間にカーソルを置き、1回クリックします。
4. ツールボックスに移動し、C1DateSelector 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。

マークアップ
<Grid x:Name="LayoutRoot"> <Xaml:C1DateSelector/> </Grid>

これで、C1DateSelector コントロールを含む UWP スタイルのアプリケーションを作成できました。次の手順では、コントロールの外観を変更します。

手順2：コントロールのカスタマイズ

前の手順では、C1DateSelector コントロールを使用して UWP スタイルの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。

次の手順に従います。

1. アプリケーションの[プロパティ]ウィンドウで、**Brush** カテゴリをクリックしてオプションを表示します。**Background**、**BorderBrush**、**Foreground** の各プロパティのカスタム色を設定できます。
   以下の XAML マークアップを `<Xaml:C1DateTimeSelector/>` タグに挿入すると、C1DateTimeSelector コントロールの外観も変更できます。

XAML マークアップ
Background="#CC96BB78" BorderBrush="#CCC7C7C7" Foreground="White"

2. コントロールの Text プロパティをカスタマイズして、**FontFamily** を Andalus に変更し、**FontSize** を 20 ピクセルに設定します。
   以下の XAML マークアップを タグに挿入すると、**FontFamily** および **FontSize** も変更できます。

XAML マークアップ
FontFamily="Andalus" FontSize="20"

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：プロジェクトの実行

手順では、C1DateSelector 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。C1DateSelector コントロールは次の図のようになります。

おめでとうございます。これで **C1DateSelector for UWP** クイックスタートは終了です。

C1DateSelector の使い方

以下のトピックでは、C1DateSelector コントロールの要素および機能について説明します。

C1DateSelector の要素

DateTimeEditors for UWP には、C1DateSelector コントロールがあります。このシンプルな日付セレクタコントロールを実行時にクリックすると、ドロップダウンリストから日付を選択できます。XAML ウィンドウに追加された C1DateSelector コントロールは、完全な機能を備えた日付セレクタになります。デフォルトでは、コントロールのインタフェースは次の図のように表示されます。

C1DateSelector コントロールは、次の要素で構成されます。

- **年セレクタ**
  ドロップダウン矢印を使用して、ドロップダウンリストから年を選択できます。
- **月セレクタ**
  ドロップダウン矢印を使用して、ドロップダウンリストから月を選択できます。
- **日付と曜日のセレクタ**
  ドロップダウン矢印を使用して、ドロップダウンリストから日付と曜日を選択できます。

C1DateSelector の外観プロパティ

C1DateSelector 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。次の表では、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、C1DateSelector 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、C1DateSelector 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
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。
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。

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、C1DateSelector 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。C1DateSelector コントロールを初めて使用される場合は、まず「**C1DateSelector 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、C1DateSelector 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。

コントロールをコード内で作成する

このトピックでは、C1DateSelector コントロールを作成し、コードを使用してコントロールに基本スタイルを適用する手順について説明します。

次の手順に従います。

1. MainPage.xaml が開いていない場合は開き、`<Grid> </Grid>` のタグセットを探します。次のようにマークアップを編集します。

マークアップ

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}" x:Name="LayoutRoot">
</Grid>
```

2. ページを右クリックし、リストから[**コードの表示**]を選択してコードビューに切り替えます。
3. 次の名前空間をアプリケーションに追加します。

▶ Visual Basic コードの書き方

```
Visual Basic
Imports C1.Xaml
```

▶ C# コードの書き方

```
C#
using C1.Xaml;
```

4. **InitializeComponent()** メソッドのすぐ下に、次のコンストラクタを挿入します。

▶ Visual Basic コードの書き方

```
Visual Basic
Dim c1dateselect1 As C1DateSelector = New C1DateSelector
```

▶ C# コードの書き方

```
C#
C1DateSelector c1dateselect1 = new C1DateSelector();
```

5. コンストラクタ内に次のコードを追加して、C1DateSelector の幅と高さを制御します。

▶ Visual Basic コードの書き方

```
Visual Basic
c1dateselect1.Height = 55
c1dateselect1.Width = 350
```

▶ C# コードの書き方

```
C#
c1dateselect1.Height = 55;
c1dateselect1.Width = 350;
```

6. C1DateSelector を表示する行をコードの最後に挿入します。

▶ Visual Basic コードの書き方

```
Visual Basic
LayoutRoot.Children.Add(c1dateselect1)
```

▶ C# コードの書き方

```
C#
LayoutRoot.Children.Add(c1dateselect1);
```

7. アプリケーションを実行します。C1DateSelector コントロールは次の図のようになります。

C1DateSelector で選択した日付を設定する

このトピックでは、XAML マークアップとコードで SelectedDate プロパティを設定する手順について説明します。

XAML の場合

次の XAML マークアップを `<Xaml:C1DateSelector/>` タグに追加して、SelectedDate プロパティを設定します。

XAML マークアップ
SelectedDate="1932, 01, 10"

コードの場合

次のコードを使用して、SelectedDate プロパティを設定します。

▶ Visual Basic コードの書き方

Visual Basic
c1dateselect1.SelectedDate = New DateTime(1932, 1, 10)

▶ C# コードの書き方

C#
c1dateselect1.SelectedDate = new DateTime(1932, 01, 10);

FirstYear プロパティを設定する

このトピックでは、XAML マークアップとコードで FirstYear プロパティを設定する手順について説明します。

XAML の場合

次の XAML マークアップを `<Xaml:C1DateSelector/>` タグに追加して、FirstYear プロパティを設定します。

XAML マークアップ
FirstYear="500"

コードの場合

次のコードを使用して、FirstYear プロパティを設定します。

▶ Visual Basic コードの書き方

Visual Basic
c1dateselect1.FirstYear = 500

▶ C# コードの書き方

C#

```
c1dateselect1.FirstYear = 500;
```

LastYear プロパティを設定する

このトピックでは、XAML マークアップとコードで LastYear プロパティを設定する手順について説明します。

XAML の場合

次の XAML マークアップを `<Xaml:C1DateSelector/>` タグに追加して、LastYear プロパティを設定します。

XAML マークアップ

```
LastYear="1950"
```

コードの場合

次のコードを使用して、LastYear プロパティを設定します。

▶ Visual Basic コードの書き方

Visual Basic

```
c1dateselect1.LastYear = 1950
```

▶ C# コードの書き方

C#

```
c1dateselect1.LastYear = 1950;
```

TimeSelector for UWP

TimeSelector for UWP 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フェースを提供します。時刻は、ドロップダウン矢印を使用するか、キーボードの矢印を使用して選択できます。

クイックスタート

このクイックスタートガイドは、**TimeSelector for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートガイドでは、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 C1TimeSelector コントロールを追加し、C1TimeSelector コントロールをカスタマイズします。

手順1：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 C1TimeSelector for UWP を使用する UWP スタイルの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**]→[**新規作成**]→[**プロジェクト**]を選択します。
2. [**新しいプロジェクト**]ダイアログボックスで、左ペインの言語を展開し、言語の下で[**Windows ストア**]を選択し、テンプレートリストで[**新しいアプリケーション (XAML)**]を選択します。名前を入力し、[**OK**]をクリックしてプロジェクトを作成します。
3. MainPage.xaml が開いていない場合は開きます。`<Grid>` タグと `<Grid>` タグの間にカーソルを置き、1回クリックします。
4. ツールボックスに移動し、C1TimeSelector 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。

マークアップ

```
<Grid x:Name="LayoutRoot">
<Xaml:C1TimeSelector/>
</Grid>
```

これで、C1TimeSelector コントロールを含む UWP スタイルのアプリケーションを作成できました。次の手順では、コントロールの外観を変更します。

手順2：コントロールのカスタマイズ

前の手順では、C1TimeSelector コントロールを使用して UWP スタイルの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。

次の手順に従います。

1. アプリケーションの[プロパティ]ウィンドウで、**Brush** カテゴリをクリックしてオプションを表示します。**Background**、**BorderBrush**、**Foreground** の各プロパティのカスタム色を設定できます。**Background** に直線グラデーションを設定することもできます。
   以下の XAML マークアップを タグに挿入すると、C1DateTimeSelector コントロールの外観も変更できます。この XAML マークアップには、`<Xaml:C1TimeSelector.Background>` タグが含まれています。

XAML マークアップ

```
<Xaml:C1TimeSelector HorizontalAlignment="Left" VerticalAlignment="Top" Margin="22,55,0,0"
Width="302" BorderBrush="#CC9CC391" Foreground="Beige">
        <Xaml:C1TimeSelector.Background>
```

```
            <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
    <GradientStop Color="#FF1C3E1F" />
              <GradientStop Color="#FF44A24C" Offset="0.869" />
            </LinearGradientBrush>
          </Xaml:C1TimeSelector.Background>
        </Xaml:C1TimeSelector>
```

2. コントロールの **Text** プロパティをカスタマイズして、**FontFamily** を Calibri、**FontSize** を 18 ピクセル、**FontWeight** を Bold に変更します。
以下の XAML マークアップを <Xaml:C1TimeSelector.Background>タグに挿入すると、**FontFamily** および **FontSize** も変更できます。

XAMLマークアップ
`FontFamily="Calibri" FontSize="18" FontWeight="Bold"`

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：プロジェクトの実行

前の手順では、C1TimeSelector 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。C1TimeSelector コントロールは次の図のようになります。

おめでとうございます。これで **C1TimeSelector for UWP** クイックスタートは終了です。

C1TimeSelector の使い方

以下のトピックでは、C1TimeSelector コントロールの要素および機能について説明します。

C1TimeSelector の要素

DateTimeEditors for UWP には、C1TimeSelector コントロールが含まれています。これは、時刻セレクタ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 C1TimeSelector コントロールは、完全な機能を備えた時刻セレクタになります。デフォルトでは、コントロールのインタフェー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。

- **時間セレクタ**
ドロップダウン矢印を使用して、ドロップダウンリストから時間を選択できます。
- **分セレクタ**
ドロップダウン矢印を使用して、ドロップダウンリストから分を選択できます。

C1TimeSelector の外観プロパティ

C1TimeSelector 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。次の表では、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、C1TimeSelector 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、C1TimeSelector 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
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。
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。

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、C1TimeSelector 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。C1TimeSelector コントロールを初めて使用される場合は、まず「C1DateSelector 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、C1TimeSelector 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。

コントロールをコード内で作成する

このトピックでは、C1TimeSelector コントロールを作成し、コードを使用してコントロールに基本スタイルを適用する手順について説明します。

次の手順に従います。

1. MainPage.xaml が開いていない場合は開き、<Grid> </Grid> のタグセットを探します。次のようにマークアップを編集します。

XAML マークアップ
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}" x:Name="LayoutRoot"> </Grid>

2. ページを右クリックし、リストから[**コードの表示**]を選択してコードビューに切り替えます。
3. 次の名前空間をアプリケーションに追加します。

▶ C# コードの書き方

C#
using C1.Xaml;

▶ Visual Basic コードの書き方

Visual Basic
Imports C1.Xaml

4. **InitializeComponent()** メソッドのすぐ下に、次のコンストラクタを挿入します。

▶ Visual Basic コードの書き方

Visual Basic
Dim c1timeselect1 As C1TimeSelector = New C1TimeSelector

▶ C# コードの書き方

C#
C1TimeSelector c1timeselect1 = new C1TimeSelector();

5. コンストラクタの下に次のコードを追加して、C1TimeSelector の幅と高さを制御します。

▶ Visual Basic コードの書き方

Visual Basic
c1timeselect1.Height = 55 c1timeselect1.Width = 350

▶ C# コードの書き方

C#
c1timeselect1.Height = 55; c1timeselect1.Width = 350;

6. C1TimeSelector を表示する行をコードの最後に挿入します。

▶ Visual Basic コードの書き方

Visual Basic

```
LayoutRoot.Children.Add(c1timeselect1)
```

C# コードの書き方

C#

```
LayoutRoot.Children.Add(c1timeselect1);
```

7. アプリケーションを実行します。C1TimeSelector コントロールは次の図のようになります。

選択されている時間を設定する

C1TimeSelector コントロールの時刻は、XAML またはコードで**SelectedTime** プロパティを設定することによって指定できます。

XAML の場合

最初に選択されている日付を指定するには、XAML マークアップを次のように編集します。

XAML マークアップ

```
<Xaml:C1DateTimeSelector SelectedTime="02:05:05"/>
```

コードの場合

Visual Basic コードの書き方

Visual Basic

```
c1timeselect1.SelectedTime = new TimeSpan( 02, 05, 05);
```

C# コードの書き方

C#

```
c1timeselect1.SelectedTime = New TimeSpan(02, 05, 05)
```

DateTimePicker for UWP

DateTimePicker for UWP を使用して、日付/時刻の情報を交換します。この機能は、日付/時刻の値、日付のみの値、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。日付/時刻は、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ガイドは、**DateTimePicker for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートガイドでは、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：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 **DateTimePicker for UWP** を使用する UWP スタイルの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**]→[**新規作成**]→[**プロジェクト**]を選択します。
2. [**新しいプロジェクト**]ダイアログボックスで、左ペインの言語を展開し、言語の下で[**Windows ストア**]を選択し、テンプレートリストで[**新しいアプリケーション (XAML)**]を選択します。**名前**を入力し、[**OK**]をクリックしてプロジェクトを作成します。
3. MainPage.xaml が開いていない場合は開きます。`<Grid>` タグと`<Grid>` タグの間にカーソルを置き、1回クリックします。
4. 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 マークアップ

```
<Grid x:Name="LayoutRoot">
<DateTimeEditors:C1DateTimePicker/>
</Grid>
```

これで、**C1DateTimePicker** コントロールを含む UWP スタイルのアプリケーションを作成できました。次の手順では、コントロールの外観を変更します。

手順2：コントロールのカスタマイズ

前の手順では、**C1DateTimePicker** コントロールを使用して UWP スタイルの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。

次の手順に従います。

1. `<DateTimeEditors:C1DateTimePicker/>` タグに `Height="60"` を追加して、コントロールの高さを決定します。XAML マークアップは次のようになります。

XAML マークアップ

```
<DateTimeEditors:C1DateTimePicker Height="60"/>
```

2. `<DateTimeEditors:C1DateTimePicker/>` タグに `Width="450"` を追加して、コントロールの幅を決定します。XAML マークアップは次のようになります。

XAML マークアップ

```
<DateTimeEditors:C1DateTimePicker Height="60" Width="450"/>
```

3. <DateTimeEditors:C1DateTimePicker/>タグに TimeFormat="ShortTime" を追加して、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。XAML マークアップは次のようになります。

```
XAML マークアップ
<DateTimeEditors:C1DateTimePicker Height="60" Width="450" TimeFormat="ShortTime"/>
```

4. <DateTimeEditors:C1DateTimePicker/> タグに DateFormat="Long" を追加して、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 マークアップ
<DateTimeEditors:C1DateTimePicker Height="60" Width="450" TimeFormat="ShortTime"
DateFormat="Long"/>
```

5. <DateTimeEditors: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 マークアップ
<DateTimeEditors:C1DateTimePicker Height="60" Width="450" TimeFormat="ShortTime"
DateFormat="Long" FirstDayOfWeek="Wednesday"/>
```

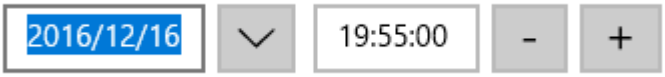
この手順では、C1DateTimePicker コントロールの外観をカスタマイズしました。次の手順では、プロジェクトを実行し、コントロールの機能を確認します。

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**]を選択します。アプリケーションは次の図のように表示されます。



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。
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 **DateTimePicker for UWP** 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 UWP スタイルの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、コントロールの実行時の外観と機能を確認しました。

C1DateTimePicker の使い方

以下のトピックでは、**C1DateTimePicker** コントロールの要素および機能について説明します。

C1DateTimePicker の要素

DateTimeEditors for UWP 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フェースは次の図のように表示されます。

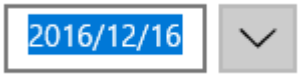
C1DateTimePicker コントロールは、次の要素で構成されます。

- **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。

C1DateTimePicker の編集モード

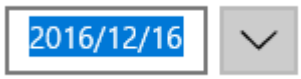
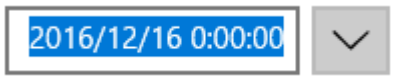
デフォルトでは、ページに配置された **C1DateTimePicker** コントロールには日付ピッカーと時刻ピッカーの両方が表示されます。表示されるピッカーは、**EditMode** プロパティを **Date**、**Time**、または **DateTime** に設定することによって変更できます。日付ピッカーのみを表示する場合は EditMode プロパティを **Date** に、時刻ピッカーのみを表示する場合は EditMode プロパティを **Time** に、日付ピッカーと時刻ピッカーの両方を表示する場合は EditMode プロパティを **DateTime** に設定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

C1DateTimePicker の日付書式

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	
Custom	

C1DateTimePicker の時刻書式

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**、**LongTime**、**TimeSpan**または**Custom** 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）	
TimeSpan (デフォルト)	
Custom (デフォルト)	

C1DateTimePicker の外観プロパティ

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。次の表

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
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。

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「**C1DateTimePicker 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 値を許可する

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 `<DateTimeEditors: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 マークアップ
<DateTimeEditors:C1DateTimePicker AllowNull="True"/>

コードの場合

次の手順に従います。

1. **MainPage.xaml.cs** ページを開きます。
2. **C1DateTimePicker_Loaded** イベントの下に次のコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic
C1DateTimePicker1.AllowNull = True

▶ C# コードの書き方

CS
c1DateTimePicker1.AllowNull = true;

3. プロジェクトを実行します。

編集モードを選択する

デフォルトでは、**C1DateTimePicker** コントロールには日付/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 `<DateTimeEditors: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 マークアップ
<DateTimeEditors: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. **MainPage.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。

   ▸ Visual Basic コードの書き方

Visual Basic
Imports C1.XAML.DateTimeEditors

   ▸ C# コードの書き方

C#
using C1.XAML.DateTimeEditors;

3. **C1DateTimePicker_Loaded** イベントの下に次のコードを追加します。

   ▸ Visual Basic コードの書き方

Visual Basic
C1DateTimePicker1.EditMode = C1DateTimePickerEditMode.Date

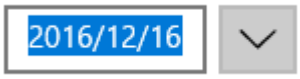
   ▸ C# コードの書き方

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。



時刻書式を選択する

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 `<DateTimeEditors: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。

マークアップ
<DateTimeEditors: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. **MainPage.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。

   ▶ Visual Basic コードの書き方

Visual Basic
Imports C1.XAML.DateTimeEditors

   ▶ C# コードの書き方

C#
using C1.WAML.DateTimeEditors;

3. **C1DateTimePicker_Loaded** イベントの下に次のコードを追加します。

   ▶ Visual Basic コードの書き方

Visual Basic
C1DateTimePicker1.TimeFormat = C1TimeEditorFormat.ShortTime

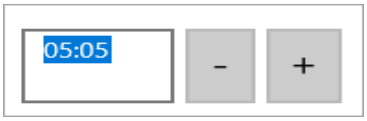
   ▶ C# コードの書き方

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。



日付書式を選択する

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 `<DateTimeEditors: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。

マークアップ
<DateTimeEditors: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. **MainPage.xaml.cs** ページを開きます。
2. **C1DateTimePicker_Loaded** イベントの下に次のコードを追加します。

   ▶ Visual Basic コードの書き方

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

   ▶ C# コードの書き方

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。

日付/時刻を指定する

C1DateTimePicker コントロールの時刻と日付を指定するには、コードを使用して **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。

コードの場合

次の手順に従います。

1. **MainPage.xaml.cs** ページを開きます。
2. **C1DateTimePicker_Loaded** イベントの下に次のコードを追加します。

   ▶ Visual Basic コードの書き方

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

   ▶ C# コードの書き方

   ```
   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/時刻が「01/17/2010 11:04:00 AM」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。

DatePicker for UWP

DatePicker for UWP を使用して、日付情報を表示および選択できます。**C1DatePicker** コントロールは、日付値を選択するための簡単で直感的な UI を提供します。C1DatePicker のドロップダウン矢印をクリックし、カレンダーから日付を選択できます。

クイックスタート

このクイックスタートガイドは、**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ガイドでは、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：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、UWP スタイルの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、左ペインの言語を展開し、言語の下で[**Windows ストア**]を選択し、テンプレートリストで[**新しいアプリケーション (XAML)**]を選択します。**名前**を入力し、[**OK**]をクリックしてプロジェクトを作成します。
3. MainPage.xaml が開いていない場合は開きます。`<Grid>` タグと `<Grid>` タグの間にカーソルを置き、1回クリックします。
4. 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 マークアップ

```
<Grid x:Name="LayoutRoot">
<DateTimeEditors:C1DatePicker></DateTimeEditors:C1DateTimePicker>
</Grid>
```

これで、C1DatePicker **for UWP** 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：コントロールのカスタマイズ

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。

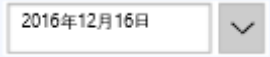
- **SelectedDateFormat** プロパティを **Long** に設定します。これで、コントロールの日付書式が変わり、月と週の完全な名前が表示されます。
- **ButtonBackground** プロパティの隣にあるドロップダウン矢印をクリックし、ドロップダウンカラーピッカーからカラーを選択します。

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 UWP スタイルの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、次の図のように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。

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 UWP 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を実行時にクリックすると、ドロップダウンカレンダーから日付を選択できます。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フェー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。

C1DatePicker の外観プロパティ

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。次の表では、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

以下のプロパティを使用すると、**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すると、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
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。

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「**C1DatePicker 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 コントロール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 値を許可する

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

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

XAML マークアップ
<DateTimeEditors: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. **C1DatePicker_Loaded** イベントの下に次のコードを追加します。

   ▶ Visual Basic コードの書き方

Visual Basic
C1DatePicker1.AllowNull = True

   ▶ C# コードの書き方

C#
c1DatePicker1.AllowNull = true;
3. プロジェクトを実行します。

日付書式を選択する

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 `<DateTimeEditors:C1DatePicker>` タグ内に配置します。マークアップは次のようになります。

XAML マークアップ
<DateTimeEditors: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. **MainPage.xaml.cs** ページを開きます。
2. **C1DateTimePicker_Loaded** イベントの下に次のコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic
C1DatePicker1.SelectedDateFormat = C1.WPF.DateTimeEditors.C1DatePickerFormat.Long

▶ C# コードの書き方

C#
c1DateTimePicker1.SelectedDateFormat = C1.WPF.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。

週の最初の曜日を設定する

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

週の最初の曜日を変更するには、`<DateTimeEditors:C1DatePicker>` タグ内に `FirstDayOfWeek="Monday"` を配置します。マークアップは次のようになります。

```
マークアップ
<DateTimeEditors: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. **C1DatePicker_Loaded** イベントの下に次のコードを追加します。

▶ Visual Basic コードの書き方

```
Visual Basic
C1DatePicker1.FirstDayOfWeek = DayOfWeek.Monday
```

▶ C# コードの書き方

```
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を設定する

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。このトピックでは、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。

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. **C1DatePicker_Loaded** イベントの下に次のコードを追加します。

▶ Visual Basic コードの書き方

```
Visual Basic
Dim dateStringStart As String = "5-2-2016"
Dim dateStringEnd As String = "25-2-2016"
c1DatePicker1.DisplayDateStart = DateTime.Parse(dateStringStart)
c1DatePicker1.DisplayDateEnd = DateTime.Parse(dateStringEnd)
```

▶ C# コードの書き方

```
C#
string dateStringStart = "5-2-2016";
string dateStringEnd = "25-2-2016";
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。

TimeEditor for UWP

TimePicker for UWP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フェースを提供します。時刻は、インクリメントボタン、デクリメントボタン、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ガイドは、**TimeEditor for UWP** を初めて使用するユーザーのために用意されています。このクイックスタートガイドでは、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：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 **TimeEditor for UWP** を使用する UWP スタイルの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**]→[**新規作成**]→[**プロジェクト**]を選択します。
2. [**新しいプロジェクト**]ダイアログボックスで、左ペインの言語を展開し、言語の下で[**Windows ストア**]を選択し、テンプレートリストで[**新しいアプリケーション (XAML)**]を選択します。**名前**を入力し、[**OK**]をクリックしてプロジェクトを作成します。
3. MainPage.xaml が開いていない場合は開きます。`<Grid>` タグと`<Grid>` タグの間にカーソルを置き、1回クリックします。
4. ツールボックスに移動し、C1TimeEditor 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 マークアップ
<Grid x:Name="LayoutRoot"> <DateTimeEditors:C1TimeEditor></DateTimeEditors:C1TimeEditor> </Grid>

これで、**TimeEditor for UWP** 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 **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、**C1TimeEditor** コントロールをカスタマイズします。

1. 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。
   - **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。
2. 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 `<DateTimeEditors:C1TimeEditor>` 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** ページを開きます。
   c. **C1TimeEditor_Loaded** イベントの下に次のコードを追加します。

▶ Visual Basic コードの書き方

```
Visual Basic
C1TimeEditor1.Value = New TimeSpan(17, 0, 0)
```

▶ C# コードの書き方

```
C#
C1TimeEditor1.Value = new TimeSpan(17,0,0);
```

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 UWP スタイルの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. ツールバーで、[**プロジェクト**]→[**プロジェクトの実行**]を選択してプロジェクトを実行します。時刻の値が「17:00」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16:00時間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 **TimeEditor for UWP** クイックスタートは終了です。

C1TimeEditor の使い方

以下のトピックでは、**C1TimeEditor** コントロールの要素および機能について説明します。

C1TimeEditor の要素

DateTimeEditors for UWP 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フェー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** プロパティを使用して設定できます。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

スピンボタンを使用し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		秒を含まない短い書式で時刻が表示されます。
LongTime（デフォルト）		秒を含む長い書式で時刻が表示されます。
TimeSpan		時間間隔が表示されます。
Custom		カスタムの時刻形式が表示されます。

C1TimeEditor の外観プロパティ

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。次の表では、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すると、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
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。

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「**C1TimeEditor 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 値を許可する

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 `<DateTimeEditors: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 マークアップ
<DateTimeEditors:C1TimeEditor AllowNull="True"/>
```

コードの場合

次の手順に従います。

1. **MainPage.xaml.cs** ページを開きます。
2. **C1TimeEditor_Loaded** イベントの下に次のコードを追加します。

   ▶ Visual Basic コードの書き方

   ```
   Visual Basic
   C1TimeEditor1.AllowNull = True
   ```

   ▶ C# コードの書き方

   ```
   C#
   c1TimeEditor1.AllowNull = true;
   ```
3. プロジェクトを実行します。

スピンボタンを削除する

ShowButtons プロパティを **Fals**e 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 `<DateTimeEditors: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 マークアップ

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

コードの場合

次の手順に従います。

1. **MainPage.xaml.cs** ページを開きます。
2. **C1TimeEditor_Loaded** イベントの下に次のコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
C1TimeEditor1.ShowButtons = False
```

▶ C# コードの書き方

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。

時刻書式を選択する

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

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 `<DateTimeEditors: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 マークアップ

```
<DateTimeEditors: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. **MainPage.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。

▶ Visual Basic コードの書き方

```
Visual Basic
Imports C1.WPF.DateTimeEditors
```

▶ C# コードの書き方

```
C#
using C1.WPF.DateTimeEditors;
```

3. **C1TimeEditor_Loaded** イベントの下に次のコードを追加します。

▶ Visual Basic コードの書き方

```
Visual Basic
C1TimeEditor1.Format = C1TimeEditorFormat.ShortTime
```

▶ C# コードの書き方

```
C#
c1TimeEditor1.Format = 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。

スピン間隔を設定する

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 マークアップ
<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. **MainPage.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。

▶ Visual Basic コードの書き方

Visual Basic
C1TimeEditor1.Interval = 1000

▶ C# コードの書き方

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。

値のインクリメント値を設定する

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<DateTimeEditors: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 マークアップ
<DateTimeEditors: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. **MainPage.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。

▶ Visual Basic コードの書き方

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

▶ C# コードの書き方

```
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。

現在の時刻を指定する

C1TimeEditor コントロールの現在の時刻は、コードで **Value** プロパティを設定することによって指定できます。

> **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 `<DateTimeEditors: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 マークアップ
<DateTimeEditors: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. **MainPage.xaml.cs** ページを開きます。
2. **C1TimeEditor_Loaded** イベントの下に次のコードを追加します。

▶ Visual Basic コードの書き方

```
Visual Basic
```

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

▶ C# コードの書き方

```
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。

時間間隔を操作する

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 07:00:00 a.m. 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。