CalendarView for WinForms

2018.04.11 更新

グレープシティ株式会社

目次

CalendarView for WinForms

ComponentOne には、**CalendarView for WinForms** が導入されています。このコントロールの機能は、日付のナビゲーションや選択にとどまりません。コントロールには、スマートタグと設計時のコレクションエディターが組み込まれているので、最小限のコード行で最大の成果を得ることができます。複数月ビュー、連続および非連続の選択、キーボードサポート、テーマオプションによって対話性と機能性がさらに高まりました。また、コントロールは、右から左のサポートを含むカルチャ設定によるグローバリゼーションをサポートしています。

さらに、既存の CalendarView コントロールの機能を拡張するコントロール **DateEdit for WinForms** も提供されています。DateEdit コントロールを使用して日付を選択および編集できます。また、マスク、データ検証、null 値、最大日/最小日、ウォーターマーク、カスタム日付書式、国際化などがサポートされます。

主な機能

CalendarView および DateEdit コントロールでは、ナビゲーション、選択、検証、日付書式、書式設定のサポートなど、数多くの機能が提供されており、開発者は、直感的で本格的な外観のアプリケーションを構築できます。

- **CalendarView の主な機能**
- **DateEdit の主な機能**

CalendarView の主な機能

- **クイックナビゲーション**
  CalendarView を使用すると、さまざまな方法で日付、月、年をすばやく簡単にナビゲーションできます。ナビゲーションボタンで、前の月または次の月に移動できます。月および年のセレクタで、それぞれのポップアップから年または月を選択できます。さらに、年フィールドを編集して、特定の年にジャンプできます。

- **選択**
  CalendarView は、単一の日付と複数の日付の選択をサポートします。複数の日付を選択する場合は、連続および非連続モードの選択が可能です。

- **複数月ビュー**
  CalendarView では、**CalendarDimensions** プロパティを設定して、カレンダーに複数の月を表示できます。

- **キーボードサポート**
  CalendarView では、ナビゲーションと選択の両方でキーボードサポートを使用できます。さまざまなキーを使用して日付と月の間を移動したり、1 か月内の複数の日付を選択することができます。

- **向き**
  CalendarView では、複数月ビューの場合に月を垂直方向にも水平方向にも表示できます。

- **国際化**
  CalendarView では、現在のカルチャ設定を変更して特定のロケールでカレンダーを表示できます。また、このコントロールは右から左のスクリプトに従う言語に対して右から左のサポートを提供します。

- **テーマ**
  CalendarView では、定義済みの**テーマ**を使用して、カレンダーのルックアンドフィールをカスタマイズできます。

- **スタイル設定**
  CalendarView には、カレンダー領域と、タイトルやナビゲーションボタンなどのカレンダー要素のスタイル設定を行うためのさまざまなスタイル設定機能があります。

DateEdit の主な機能

- **日付書式**
  DateEdit を使用すると、短い日付、長い日付、標準の日付、カスタム日付書式など、定義済みの書式で日付を表示できます。

- **データ検証**
  DateEdit は、入力文字列自体を検証する事前検証と、エンドユーザーによって入力された値を検証する事後検証という 2 種類のデータ検証をサポートします。

- **null 値**
  DateEdit では、読み取り専用モードと編集モードのどちらのモードでも、null 値の処理に関して柔軟な規則が提供されています。

- **書式設定と解析**
  DateEdit では、標準およびカスタムの書式指定子を使用できるほか、値を書式設定することもできます。

- **マスク**
  DateEdit では、ユーザーの入力を制限したり無効な文字を防ぐためのマスクがサポートされています。

- **国際化**
  DateEdit は国際化をサポートします。このコントロールは、変更を加えることなくさまざまな言語やカルチャに対応でき、右から左への筆記法に従う言語もサポートしています。

- **スタイル設定**
  DateEdit は、タイトルやナビゲーションボタンなどの要素のスタイル設定をカスタマイズする機能を備えています。

先頭に戻る

オブジェクトモデルの概要

CalendarView コントロールおよびDateEdit コントロールには、さまざまなクラス、オブジェクト、関連するメソッドおよびプロパティを提供するリッチオブジェクトモデルが用意されています。このセクションでは、CalendarView コントロールとDateEdit コントロールのオブジェクトモデルの概要について個別に説明します。

- CalendarView
- DateEdit

CalendarView

C1CalendarView
プロパティ: AnnuallyBoldedDates, **BackColor**, **BackgroundImage**, **BackgroundImageLayout**, **BoldedDates**, **CalendarDimensions**, **CalendarWeekRule**, **CurrentMonthDisplayOffset**, **DayTitlePosition, DisabledDates, FirstDayOfWeek**, **ForeColor**, **MaxColumns**, **MaxDate**, **MaxSelectionCount**, **MinDate**, **MonthTitlePosition**, **PeriodSelectionType**, **RightToLeftLayout**, **SelectedDates**, **ShowArrowButtons**, **ShowToday**, **ShowToolTips**, **ShowWeekNumbers**, **Theme**, **VerticalOrientationLayout**, **WorkDays** **イベント:RightToLeftLayoutChanged**, **SelectionChanged**, **StylesChanged**
BaseArrowStyle
プロパティ: BackImage, **BackImageAlignment**, **BackImageScaling**, **ForeColor**
BaseStyle
プロパティ: BackColor, **Border**, **BorderColor**, **HorizontalAlignment**, **Name**, **VerticalAlignment**
CalendarTheme
プロパティ: Common, **Day**, **NavigationButtons**, **Titles**
CommonStyle
プロパティ: BackImage, **BackImageAlignment**, **BackImageScaling**, **VerticalAlignment**
DayStyle
プロパティ: Font, **ForeColor**
DayTheme
プロパティ: Bolded, **Disabled**, **Ordinary**, **Selected**, **Today**, **Trail**, **Weekend**
DayTitleStyle
プロパティ: BackImage, **BackImageAlignment**, **BackImageScaling**, **ForeColor**
MonthTitleStyle
プロパティ: BackImage, **BackImageAlignment**, **BackImageScaling**, **ForeColor**, **Padding**, **Trimming**
NavigationButtonsTheme
プロパティ: ArrowNext, **ArrowPrevious**, **ImageArrowNext**, **ImageArrowPrevious**
TitleTheme
プロパティ: Day, **Month**, **Week**, **Weekend**
WeekTitleStyle
プロパティ: Font, **ForeColor**

先頭に戻る

DateEdit

C1DateEdit
プロパティ: AllowSpinLoop, **Calendar**, **FormatType**
CalendarSettings
プロパティ: AnnuallyBoldedDates, **ArrowColor**, **BackColor**, **BoldedDates**, **CalendarDimensions**, **CalendarWeekRule**, **CaptionFormat**, **ClearText**, **CurrentMonthDisplayOffset**, **DateIsNull**, **DayNameLength**, **DayNamesColor**, **DayNamesFont**, **DisabledDates**, **FirstDayOfWeek**, **FirstMonth**, **Font**, **ForeColor**, **LastMonth**, **LineColor**, **MaxDate**, **MinDate**, **RightToLeft**, **RightToLeftLayout**, **SelectedDate**, **SelectionBackColor**, **SelectionForeColor**, **ShowClearButton**, **ShowToday**, **ShowTodayButton**,

ShowTodayCircle, **ShowWeekNumbers**, **TitleBackColor**, **TitleFont**, **TitleForeColor**, **TitleHeight**, **TitleNavigation**, **TodayBorderColor**, **TodayText**, **TrailingForeColor**, **VisualStyle** **イベント: ClearButtonClick**, **ClearButtonVisibilityChanged**, **DateValueChanged**, **DateValueSelected**, **MonthChanged**, **RightToLeftLayoutChanged**, **TodayButtonClick**, **TodayButtonVisibilityChanged**, **VisualStyleChanged**
DropDownCalendar
プロパティ: BackColor, **Font**, **ForeColor**

先頭に戻る

クイックスタート

このクイックスタートでは、CalendarView コントロールと DateEdit コントロールの使用に取りかかります。そのために、WinForms アプリケーションを作成し、それに CalendarView コントロールと DateEdit コントロールを追加し、CalendarView から DateEdit に日付を入力します。

このクイックスタートでは、CalendarView を DateEdit に連結して日付を入力します。ただし、DateEdit のカレンダーポップアップから DateEdit に日付を直接入力することもできます。

コントロールの使用をすぐに開始するには、次の手順に従います。

1. **CalendarView および DateEdit コントロールをアプリケーションに追加する**
2. **CalendarView および DateEdit コントロールをカスタマイズする**
3. **CalendarView から DateEdit に日付を入力するコードを追加する**

次の図は、CalendarView コントロールから今日の日付を選択して DateEdit コントロールに表示したところです。

手順 1 :CalendarView および DateEdit コントロールをアプリケーションに追加する

1. Visual Studio で **Windows フォームアプリケーション**を作成します。
2. **C1CalendarView** コントロールをツールボックスからアプリケーションにドラッグアンドドロップします。
3. **C1DateEdit** コントロールをツールボックスからアプリケーションにドラッグアンドドロップします。

手順 2:CalendarView および DateEdit コントロールをカスタマイズする

1. コードビューに切り替えます。
2. 次のコードを追加して、C1Calendarview および DateEdit の場所、サイズ、背景色をカスタマイズします。
    - **Visual Basic**

```
' CalendarViewの位置を設定します
C1CalendarView1.Location = New System.Drawing.Point(300, 152)

' CalendarViewの背景色を設定します
C1CalendarView1.Theme.Common.BackColor =
    System.Drawing.Color.FromArgb(CInt(CByte(224)),
                                  CInt(CByte(224)),
```

```
                                        CInt(CByte(224)))

    ' CalendarViewを非表示にします
    C1CalendarView1.Visible = False

    ' DateEditのサイズと位置を設定します
    C1DateEdit1.Size = New System.Drawing.Size(150, 20)
    C1DateEdit1.Location = New System.Drawing.Point(330, 126)
```

- **C#**

```
    // CalendarViewの位置を設定します
    // c1CalendarView1.Location = new System.Drawing.Point(300, 152);

    // CalendarViewの背景色を設定します
    c1CalendarView1.Theme.Common.BackColor = System.Drawing.Color.FromArgb
                                                (((int)(((byte)(224)))),
                                                ((int)(((byte)(224)))),
                                                ((int)(((byte)(224)))));
    // CalendarViewを非表示にします
    c1CalendarView1.Visible = false;

    // DateEditのサイズと位置を設定します
    // c1DateEdit1.Size = new System.Drawing.Size(150, 20);
    c1DateEdit1.Location = new System.Drawing.Point(330, 126);
```

先頭に戻る

手順 3: CalendarView から DateEdit に日付を入力するコードを追加する

次のコードを追加して、DateEdit をクリックすると CalendarView がポップアップ表示され、CalendarView から日付を選択して DateEdit コントロールに入力できるようにします。

- **Visual Basic**

```
Private Sub Form1_Load(sender As Object, e As EventArgs)
    ' テキストボックスのClickイベントを処理します
    AddHandler c1DateEdit1.Click, AddressOf c1DateEdit1_Click

    ' CalendarViewのSelectionChangedイベントを処理します
    AddHandler c1CalendarView1.SelectionChanged, AddressOf CalendarView_SelectionChanged

    c1CalendarView1.Visible = False

    ' DateEditのカレンダーポップアップを非表示にします
    c1DateEdit1.ShowDropDownButton = False
End Sub
Private Sub c1DateEdit1_Click(sender As Object, e As EventArgs)
    c1CalendarView1.Visible = True
    AddHandler c1CalendarView1.SelectionChanged, AddressOf CalendarView_SelectionChanged
End Sub

Private Sub CalendarView_SelectionChanged(sender As Object, e As EventArgs)
    ' CalendarViewから選択した日付をテキストボックスに入力します
    If c1CalendarView1.Visible Then
        c1DateEdit1.Text = c1CalendarView1.SelectedDates(0).ToShortDateString()
    End If

    ' CalendarViewを非表示にします
    c1CalendarView1.Hide()
End Sub
```

- **C#**

```
private void Form1_Load(object sender, EventArgs e)
{
    // テキストボックスのClickイベントを処理します
    c1DateEdit1.Click += c1DateEdit1_Click;
```

```
        // CalendarViewのSelectionChangedイベントを処理します
        c1CalendarView1.SelectionChanged += CalendarView_SelectionChanged;

        c1CalendarView1.Visible = false;

        // DateEditのカレンダーポップアップを非表示にします
        c1DateEdit1.ShowDropDownButton = true;
    }
    private void c1DateEdit1_Click(object sender, EventArgs e)
    {
        //c1CalendarView1.Visible = true;
        //c1CalendarView1.SelectionChanged += CalendarView_SelectionChanged;
    }

    private void CalendarView_SelectionChanged(object sender, EventArgs e)
    {
        // CalendarViewから選択した日付をテキストボックスに入力します
        if (c1CalendarView1.Visible)
            c1DateEdit1.Text = c1CalendarView1.SelectedDates[0].ToShortDateString();

        // CalendarViewを非表示にします
        c1CalendarView1.Hide();
    }
```

先頭に戻る

要素

CalendarView は、日、月、年をナビゲーションしたり、カレンダーのタイトルを表示したり、カスタムの日付を作成するためのさまざまな要素で構成されます。

以下のセクションでは、CalendarView の要素について説明します。

ナビゲーションとタイトルの要素
: CalendarView のナビゲーションとタイトルの要素について学習します。

月ビューの要素
: CalendarView の月ビューの要素について学習します。

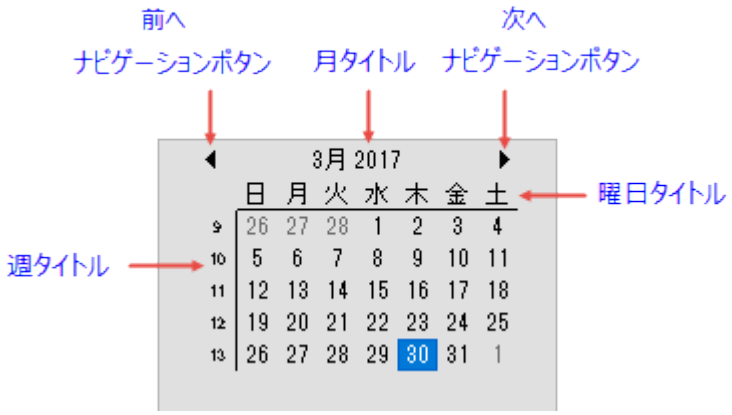
DateEdit の要素
: DateEdit のドロップダウン、上下スピンボタン、コンボボックス、カレンダーポップアップなどの要素について学習します。

ナビゲーションとタイトルの要素

CalendarView には、次のナビゲーション要素とタイトル要素があります。

- **ナビゲーションボタン**：CalendarView には、それぞれ前と次の月に移動するための[前へ]および[次へ]ナビゲーションボタンがあります。
- **月および年ポップアップセレクタ**：CalendarView には、月タイトルをクリックすると表示される月および年ポップアップセレクタがあります。これらのセレクタのリストから月または年を選択して、それらの月または年にジャンプできます。
- **タイトル**：CalendarView には、次のタイトルが含まれます。
  - **月タイトル**：月タイトル（またはカレンダータイトル）は、デフォルトでカレンダーの上部に表示され、月と年を表示します。
  - **週タイトル**：週タイトルは、カレンダーの左に表示され、週番号を示します。
  - **曜日タイトル**：曜日タイトルは、月タイトルの下に表示され、曜日名を示します。

次の図に、カレンダーナビゲーションシステムを構成する要素を示します。

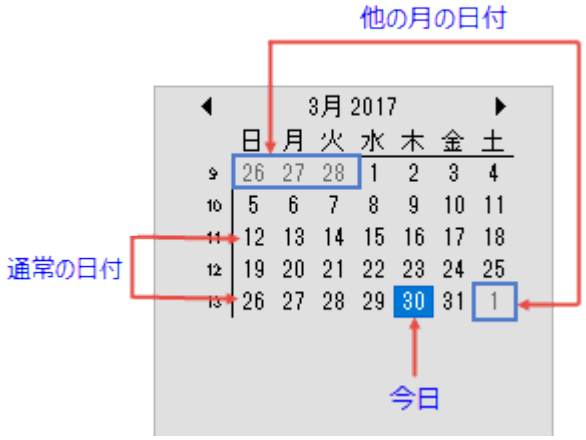


月ビューの要素

CalendarView には、次の月ビュー要素があります。

- **通常の日付**:現在の月に属する日付です。
- **選択された日付**:設計時または実行時のいずれかに選択された日付です。
- **今日の日付**:今日の日付です。日付の周囲が四角形で塗りつぶされます。
- **無効な日付**:選択できない日付です。
- **他の月の日付**:現在の月の最初または最後の週と同じ週にあるが、前または次の月に属する日付です。

次の図に、CalendarView の月ビュー領域を構成する要素を示します。



DateEditの要素

DateEdit は次の要素で構成されます。

- **日時コンボボックス**:DateEdit には日時コンボボックスがあり、日付を手動入力することも、カレンダーポップアップから日付を選択して入力することもできます。
- **アップ/ダウンスピンボタン**:DateEdit には、選択された部分(日、月、年、時、分、秒)の値を 1 増減させるアップ/ダウンスピンボタンがあります。
- **ドロップダウンボタン**:DateEdit にはドロップダウンボタンがあり、これをクリックすると、カレンダーポップアップが表示されます。
- **カレンダーポップアップ**:DateEdit にはカレンダーポップアップがあり、タイトル、ナビゲーションボタン、月および年ポッ

プアップセレクタ、月ビューの要素、および[今日]ボタンと[クリア]ボタンで構成されています。

次の図に、DateEdit コントロールを構成する要素を示します。

設計時サポート

CalendarView および DateEdit には、オブジェクトモデルの操作を簡略化する設計時サポートを提供します。どちらのコントロールにもプロパティに簡単にアクセスできるスマートタグが含まれていますが、CalendarViewにはコレクションエディタがあり、日付の追加と削除を迅速に行うことができます。

スマートタグ
: スマートタグを使用して、よく使用される CalendarView のプロパティにアクセスする方法を学習します。

コレクションエディター
: コレクションエディターを使用して、さまざまなタイプの日付を追加または削除する方法を学習します。

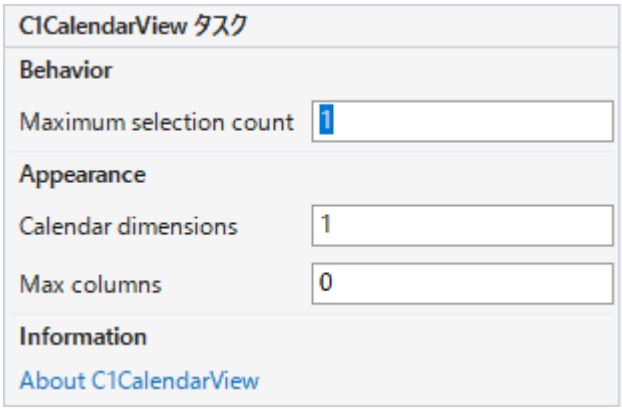
スマートタグ

CalendarView と DateEdit では、オブジェクトモデルの操作を簡略化する設計時サポートが提供されています。どちらのコントロールにも、よく使用するプロパティなどのオプションに簡単にアクセスできるスマートタグが含まれています。スマートタグは、[C1CalendarView タスク]メニュー、および[C1DateEdit タスク]メニューへのショートカットで、コントロールでよく使用されるプロパティにすばやくアクセスできます。

このセクションでは、CalendarView コントロールと DateEdit コントロールそれぞれのタスクメニューへのアクセスに使用されるスマートタグについて説明します。

- **CalendarView のタスクメニュー**
- **DateEdit のタスクメニュー**

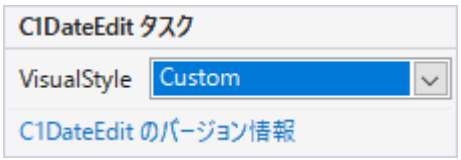
CalendarView のタスクメニュー



[C1CalendarView タスク]メニューには、次のオプションがあります。

- **最小日**
  使用できる最小の日付を設定できます。
- **最大日**
  使用できる最大の日付を設定できます。
- **最大選択数**
  選択できる日付の最大数を設定できます。
- **カレンダーサイズ**
  表示する月の数を設定できます。
- **最大列**
  カレンダーサイズに対する最大の列数を設定できます。
- **C1CalendarView のバージョン情報**
  コントロールとオンラインリソースのバージョン番号を確認できます。

DateEdit のタスクメニュー



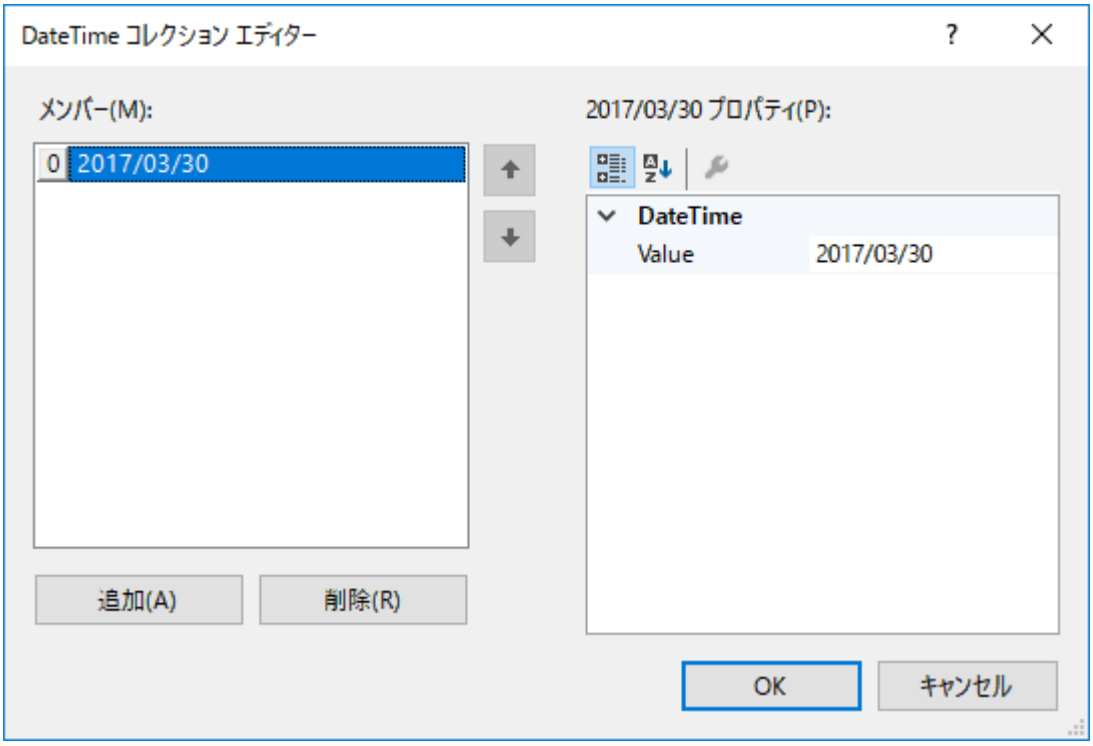
[C1DateEdit タスク]メニューには、次のオプションがあります。

- **VisualStyle**
  DateEdit コントロールの事前定義されたテーマから好みのテーマを設定できます。

- **C1DateEdit のバージョン情報**
  コントロールとオンラインリソースのバージョン番号を確認できます。

コレクションエディター

CalendarView の DateTime コレクションエディターを使用して、毎年太字の日付、太字の日付、無効な日付を追加または削除することができます。



DateTime コレクションエディターにアクセスして毎年太字の日付を追加または削除するには、次の手順に従います。

1. **C1CalendarView** コントロールを右クリックし、コンテキストメニューから**[プロパティ]**を選択します。
2. **[プロパティ]**ウィンドウで、**AnnuallyBoldedDates** プロパティの隣にある省略符ボタンをクリックして、DateTime コレクションエディターを開きます。

メモ:太字の日付または無効な日付の DateTime コレクションエディターにアクセスするには、それぞれ **BoldedDates** または **DisabledDates** プロパティの隣にある省略符ボタンをクリックします。

コントロールの使用

以下の各セクションでは、CalendarView コントロールと DateEdit コントロールで提供されているさまざまな機能について学習します。

CalendarView の機能
: CalendarView コントロールを使用して、マウスおよびキーボードで日付、月、または年を移動したり選択する方法を学習します。

DateEdit の機能
: DateEdit コントロールを使用して、いくつかの定義済み書式またはカスタム書式で日付を入力または表示する方法を学習します。

CalendarViewの機能

CalendarView には、日、月、年の間を移動したり選択するための機能があります。さらに、カルチャを設定したり、複数の月を表示したり、カレンダーの外観をカスタマイズするための機能もサポートしています。

以下のセクションでは、CalendarView のさまざまな機能について説明します。

ナビゲーション
: CalendarView で日、月、年の間を移動する方法を学習します。

選択

CalendarView で単一の日または複数の日を選択する方法を学習します。

キーボードサポート
CalendarView でキーの組み合わせを使用してすばやく移動および選択する方法を学習します。

複数月ビュー
CalendarView で複数の月を表示する方法を学習します。

カスタムの日付
CalendarView でカスタムの日付を追加または削除する方法を学習します。

カルチャ設定
CalendarView で現在のカルチャを変更する方法を学習します。

右から左のサポート
CalendarView で右から左へのサポートを実装する方法を学習します。

外観とスタイル
CalendarViewの外観をカスタマイズする方法を学習します。

ナビゲーション

CalendarView には、日、月、年の間を移動する高度なナビゲーションシステムがあります。次の 3 つの方法でナビゲーションできます。

- **ナビゲーションボタン**
- **月と年のポップアップセレクタ**
- **年フィールド**

ナビゲーションボタン

CalendarView には、[前へ]および[次へ]のナビゲーションボタンがあり、最大および最小日付範囲内で、前の月または次の月に移動できます。別の月に移動すると、前の月または次の月で選択可能な最初の日が現在の日に設定されます。

次の図に、ナビゲーションボタンを示します。

月と年のポップアップセレクタ

カレンダータイトルの月をクリックすると、ポップアップ月セレクタが開きます。セレクタには、現在の月の 4 か月前から 4 か月後までの 9 か月が表示されます。同様に、カレンダータイトルの年をクリックすると、年セレクタがポップアップして 9 年が表示されます。ここには、現在の年の前 4 年間と後 4 年間が含まれます。ポップアップ年セレクタは、**PeriodSelectionType** プロパティの値を **List** に設定した場合にのみ開きます。

次の図に、年と月のポップアップセレクタを示します。

ポップアップ月セレクタ

ポップアップ年セレクタ

先頭に戻る

年フィールド

PeriodSelectionType が **Field** に設定されている場合は、カレンダータイトルの年をクリックすると、年フィールドがアクティブになります。この年フィールドに年の値を入力して、現在の年から特定の年にジャンプできます。ただし、入力値は、選択可能な日付の範囲内になくてはなりません。

次の図に、年フィールドを示します。

先頭に戻る

選択

CalendarView では、選択できる最大日数に応じて、1 日または複数の日を選択できます。ただし、コントロールのデフォルトでは、1 日だけを選択できます。選択可能な日の最大数を指定するには、**MaxSelectionCount** プロパティを設定します。

複数の選択

CalendarView で複数の日を選択する場合は、連続または非連続の選択がサポートされます。連続選択では、[Shift]キーを押しながら、隣接する日を選択できます。非連続の選択では、[Ctrl]キーを押しながら、個別の日を選択できます。

選択可能な日

選択可能な日は、選択を行うことができる日です。無効に設定されておらず、使用できる最小の日付と最大の日付の間が選

択可能な日になります。デフォルトでは、使用できる最小の日付と最大の日付は、それぞれ 9999/12/31 と 0001/01/01 です。使用できる最小の日付と最大の日付を指定するには、それぞれ **MaxDate** と **MinDate** プロパティを設定します。

週番号セレクタ

CalendarView には、カレンダーの左に垂直方向に表示される週番号セレクタが用意されています。このセレクタには、カレンダー月内の各週を表す週番号が表示されます。カレンダーは 1 年で 52 週あるので、CalendarView には 1 ～ 52 の番号があります。

週番号セレクタを使用すると、クリック 1 つで特定の週のすべての日を選択できます。ただし、週のすべての日を選択できるのは、MaxSelectionCount が 7 以上の場合に限ります。

キーボードサポート

CalendarView は、日や月をすばやく移動および選択するためのキーボードサポートを提供しています。次の表に、適用できるキーの組み合わせと対応するアクションを示します。

キーボードコマンド	アクション
↑	1 つ上の位置にある日に移動または選択します。
↓	1 つ下の位置にある日に移動または選択します。
←	1 つ左の位置にある日に移動または選択します。
→	1 つ右の位置にある日に移動または選択します。
Home	現在の月の最初日に移動します。
End	現在の月の最後日に移動します。
Page Up	前の月の同じ日に移動します。
Page Down	次の月の同じ日に移動します。
Shift + ↑	選択可能な最大日に基づき、現在の日から前方に連続する日を選択します。
Shift + ↓	選択可能な最大日に基づき、現在の日から後方に連続する日を選択します。
Shift + ←	最大の選択可能な日まで、左方向に 1 つずつ連続した日を選択します。
Shift + →	最大の選択可能な日まで、右方向に 1 つずつ連続した日を選択します。
Shift + Home	選択可能な最大日に基づき、現在の日から前方に連続する日を選択します。
Shift + End	選択可能な最大日に基づき、現在の日から後方に連続する日を選択します。
Shift + Page Up	選択可能な最大日に基づき、現在の日から前方に連続する日を選択します。
Shift + Page Down	選択可能な最大日に基づき、現在の日から後方に連続する日を選択します。
Ctrl + ↑	最大の選択可能な日に基づいて、現在の日から同じ列にある上方向の日を選択します。
Ctrl + ↓	最大の選択可能な日に基づいて、現在の日から同じ列にある下方向の日を選択します。

Ctrl + ←	最大の選択可能な日まで、左方向に 1 つずつ連続した日を選択します。
Ctrl + →	最大の選択可能な日まで、右方向に 1 つずつ連続した日を選択します。
Ctrl + Home	選択している現在の日を維持したまま、現在の月の最初の日を選択します。
Ctrl + End	選択している現在の日を維持したまま、現在の月の最後の日を選択します。
Ctrl + Page Up	選択している現在の月の現在の日を維持したまま、前の月の同じ日を選択します。
Ctrl + Page Down	選択している現在の月の現在の日を維持したまま、次の月の同じ日を選択します。

複数月ビュー

CalendarView は複数月ビューをサポートしています。複数月ビューを使用すると、同時に同じ画面で複数の月カレンダーを表示して、丸 1 年間のビューを作成できます。年間ビューは一覧性がよく、理解が容易です。また、ユーザーはナビゲーションや選択をシームレスに行うことができます。さらに、1 年間を一目で確認できるため、長期的な活動やプロジェクトの計画が容易になります。たとえば、複数月モードですべての月を表示すると、特定の年の祝日を簡単に確認できます。

複数月表示

CalendarView では、高速なレンダリングで複数月ビューを表示できます。カレンダー領域に、複数の月を表示するように簡単に設定できます。

CalendarView で複数の月を表示するには、**CalendarDimensions** プロパティを 2 以上に設定し、コントロールのサイズを変更します。コントロールは、CalendarDimensions で設定した値に応じて、有効なスペースに可能な限り多くの月を表示します。

水平方向と垂直方向の表示

CalendarView は、水平方向の表示と垂直方向の表示をサポートします。水平方向のカレンダーには、行より多くの列が含まれ、垂直方向のカレンダーには、列より多くの行が含まれます。

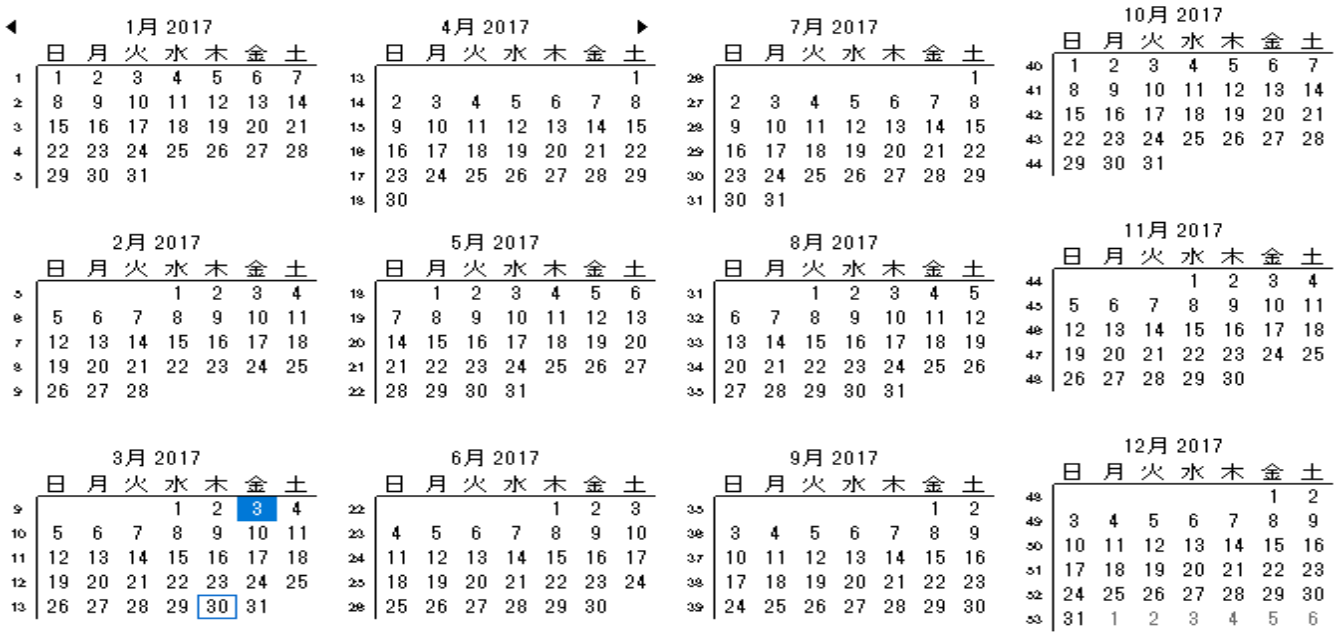
CalendarView を水平方向または垂直方向に表示するには、コントロールの高さを減らして幅を増やすか、その逆を行います。コントロールに表示する列の数に応じて幅を設定します。表示する列の数を指定するには、**MaxColumns** プロパティを設定します。

向き

CalendarView は、水平方向と垂直方向の両方をサポートします。水平方向がデフォルトの方向です。

CalendarView コントロールには、水平方向では左から右に、垂直方向では上から下に複数の月が表示されます。複数月ビューを垂直方向に表示するには、**VerticalOrientationLayout** プロパティを設定します。

次の図に、CalendarView の複数月ビューを使用して、2017 年の年間カレンダーを示します。



次のコードスニペットでは、CalendarView で複数月ビューを実装する方法を示しています。

- **Visual Basic**

```
' 表示にする月数を指定します
C1CalendarView1.CalendarDimensions = 12

' 表示にする列の最大数を指定します
C1CalendarView1.MaxColumns = 4

' 縦方向を設定します
C1CalendarView1.VerticalOrientationLayout = True
```

- **C#**

```
// 表示にする月数を指定します
c1CalendarView1.CalendarDimensions = 12;

// 表示にする列の最大数を指定します
c1CalendarView1.MaxColumns = 4;

// 縦方向を設定します
c1CalendarView1.VerticalOrientationLayout = true;
```

カスタムの日付

CalendarView は、選択された日付、太字の日付、毎年太字の日付、無効な日付などのカスタムの日付をサポートします。カスタムの日付を使用すると、重要なイベントがある日付を強調表示したり、カレンダーにカスタム機能を提供するために特定の日付を無効にすることができます。コントロールの DateTime コレクションエディターおよびコードからカスタムの日付を追加または削除できます。

以下のセクションでは、CalendarView でカスタムの日付を追加する方法について説明します。

選択された日付
: CalendarView でコードから選択された日付を追加する方法を学習します。

太字の日付
: CalendarView でコードから太字の日付を追加する方法を学習します。

無効な日付

CalendarView でコードから無効な日付を追加する方法を学習します。

選択された日付

CalendarView では、選択された日付は、異なる背景色と前景色で強調表示されます。

CalendarView に選択された日付を追加するには、**C1CalendarView** の **SelectedDates** プロパティを設定します。

次の図に、CalendarView の選択された日付を示します。

次のコードスニペットは、CalendarView の SelectedDates プロパティを使用して、選択された日付を追加する方法を示します。

- **Visual Basic**

```
' 選択した日付を追加します
C1CalendarView1.SelectedDates =
    New DateTime() {New DateTime(2017, 2, 23, 0, 0, 0, 0)}
```

- **C#**

```
// 選択した日付を追加します
c1CalendarView1.SelectedDates = new DateTime[]
{
    new DateTime(2017, 2, 23, 0, 0, 0, 0)
};
```

太字の日付

太字の日付を使用すると、カレンダーの特定の日付にある重要なイベント、予定、活動を強調表示できます。

設計時には、DateTime コレクションエディターを使用して太字の日付を追加できます。DateTime コレクションエディターを使用して太字の日付を追加する方法の詳細については、「コレクションエディター」を参照してください。CalendarView にコードで太字の日付を追加するには、**BoldedDates** プロパティを設定します。

次の図に、CalendarView の太字の日付を示します。

次のコードスニペットは、CalendarView の BoldedDates プロパティを使用して、太字の日付を追加する方法を示します。

- **Visual Basic**

```
' 太字の日付を追加します
C1CalendarView1.BoldedDates =
    New DateTime() {New DateTime(2017, 3, 13, 0, 0, 0, 0)}
```

- **C#**

```
// 太字の日付を追加します
c1CalendarView1.BoldedDates = new DateTime[]
{
    new DateTime(2017, 3, 13, 0, 0, 0, 0)
};
```

無効な日付

無効な日付は、実行時に無効状態で表示され、選択できません。

設計時には、DateTime コレクションエディターで無効な日付を追加します。DateTime コレクションエディターを使用して無効な日付を追加する方法の詳細については、「コレクションエディター」を参照してください。CalendarView にコードで無効な日付を追加するには、**DisabledDates** プロパティを設定します。

次の図に、CalendarView の無効な日付を示します。

次のコードスニペットは、CalendarView の DisabledDates プロパティを使用して、無効な日付を追加する方法を示します。

- **Visual Basic**

```
' 無効な日付を追加します
C1CalendarView1.DisabledDates =
    New DateTime() {New DateTime(2017, 3, 20, 0, 0, 0, 0)}
```

- **C#**

```
// 無効な日付を追加します
c1CalendarView1.DisabledDates = new DateTime[]
{
    new DateTime(2017, 3, 20, 0, 0, 0, 0)
};
```

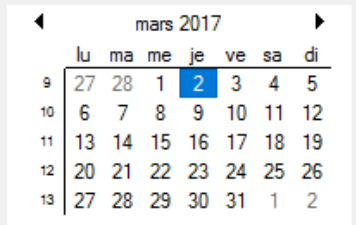
カルチャ設定

CalendarView の地域設定を変更して、特定のロケールまたは言語でカレンダーを表示できます。コントロールのカルチャ設定に基づいて、データの書式設定、解析、検証を行うことができます。カルチャ設定には、文字列比較、数値や日時の書式、小数点文字などの特殊文字を含めることができます。

CalendarView では、System.Globalization 名前空間を通じて、週タイトルと月タイトルを完全にローカライズできます。この名前空間には、言語、国/地域、書式パターン(日付、通貨、数値)などのカルチャに関する情報が含まれます。現在のカルチャのロケールを指定すると、週タイトルと月タイトルに表示される文字列に影響します。デフォルトでは、現在のカルチャは、**System.Threading.Thread.CurrentThread** の CurrentCulture プロパティから指定されます。

現在のカルチャ以外のカルチャを使用するには、カルチャ名を指定して、目的のカルチャを設定します。さまざまなカルチャ名と識別子については、.NET Framework ドキュメントの CultureInfo クラスを参照してください。

次の図に、フランス語カルチャが適用された CalendarView を示します。



次のコードスニペットは、CalendarView で現在のカルチャを設定する方法を示します。

- **Visual Basic**

```
' 目的のカルチャーを設定します。この例では、フランス語(フランス)ロケールに設定しています
System.Threading.Thread.CurrentThread.CurrentCulture = New System.Globalization.CultureInfo("fr-FR")

' この呼び出しは、Windowsフォームデザイナーによって必要とされます
InitializeComponent()
```

- **C#**

```
// 目的のカルチャーを設定します。この例では、フランス語(フランス)ロケールに設定しています
System.Threading.Thread.CurrentThread.CurrentCulture = new System.Globalization.CultureInfo("fr-FR");

// この呼び出しは、Windowsフォームデザイナーによって必要とされます
InitializeComponent();
```

右から左のサポート

CalendarView は、右から左への筆記法に従う右から左の機能をサポートします。

このコントロールでは、**RightToLeftLayout** および **RightToLeft** プロパティを使用して、カレンダーを右から左の方向で表示できます。RightToLeftLayout と RightToLeft のプロパティをそれぞれ **true** と **Yes** に設定すると、カレンダーは右から左の方向で表示されます。

次の図に、右から左の方向の CalendarView を示します。

次のコードスニペットは、CalendarView で RightToLeftLayout および RightToLeft プロパティを設定する方法を示します。

- **Visual Basic**

```
' 右から左へのレイアウトを有効にします
C1CalendarView1.RightToLeftLayout = True

' RightToLeftプロパティを設定します
C1CalendarView1.RightToLeft = RightToLeft.Yes
```

- **C#**

```
// 右から左方向のレイアウトを有効にします
c1CalendarView1.RightToLeftLayout = true;

// RightToLeftプロパティを設定します
c1CalendarView1.RightToLeft = RightToLeft.Yes;
```

外観とスタイル設定

CalendarView では、カレンダーだけでなく、月ビュー、ナビゲーション、タイトルなどの要素もカスタマイズできる、さまざまなスタイル設定機能が提供されています。

提供されているスタイル設定機能は次のとおりです。

- **配置**:カレンダー、およびカレンダーのタイトル、ナビゲーション、月ビューなどの要素に水平または垂直方向の配置を適用できます。
  - **水平方向の配置**:カレンダーまたは要素を水平方向に配置するには、**HorizontalAlignment** プロパティを使用します。
  - **垂直方向の配置**:カレンダーまたは要素を垂直方向に配置するには、**VerticalAlignment** プロパティを使用します。

- **背景画像**:カレンダー、ナビゲーションボタン、タイトル(日タイトル、月タイトル、週末タイトルなど)に背景画像を追加、配置し、画像を拡大縮小できます。
  - **背景画像**:背景画像を設定するには、**BackImage** プロパティを使用します。
  - **背景画像の配置**:背景画像の配置を設定するには、**BackImageAlignment** プロパティを使用します。
  - **背景画像の拡大縮小**:背景画像を拡大縮小するには、**BackImageScaling** プロパティを使用します。

- **よく使用されるスタイル**:境界線、境界線の色、フォント設定、背景色、前景色など、よく使用されるスタイルをカレンダーおよびカレンダー内の要素に適用できます。
  - **境界線**:ビジュアル要素の幅を設定するには、**Border** プロパティを設定します。
  - **境界線の色**:境界線の色を変更するには、**BorderColor** プロパティを設定します。
  - **フォント設定**:太字の日付、通常の日付、選択された日付、今日の日付、前後の月の日付、週末の日付など、あらゆる種類の日付、および日、月、週、週末などのすべてのタイトルにフォント設定を適用できます。
  - **背景色**:カレンダーまたは要素の背景色を変更するには、**BackColor** プロパティを設定します。
  - **前景色**:カレンダー内のさまざまな日付、タイトル、ナビゲーションボタンに前景色を適用するには、**ForeColor** プロパティを設定します。

- **ナビゲーションボタンの画像**:[戻る]および[次へ]ナビゲーションボタンの画像を設定できます。
  - **[戻る]矢印ボタンの画像**:[戻る]矢印ボタンの画像を設定するには、**ImageArrowPrevious** プロパティを設定します。
  - **[次へ]矢印ボタンの画像**:[次へ]矢印ボタンの画像を設定するには、**ImageArrowNext** プロパティを設定します。

次の図は、CalendarView およびその要素にスタイル設定機能を適用したところです。

次のコードスニペットは、関連するプロパティを使用して CalendarView でスタイル設定機能を実装する方法を示します。

- **Visual Basic**

```
' タイトルおよびカレンダーの水平方向と垂直方向の配置を設定します
C1CalendarView1.Theme.Titles.Week.HorizontalAlignment = C1.Framework.Alignment.Far
C1CalendarView1.Theme.Common.HorizontalAlignment = C1.Framework.Alignment.Near
C1CalendarView1.Theme.Titles.Week.VerticalAlignment = C1.Framework.Alignment.Far
C1CalendarView1.Theme.Common.VerticalAlignment = C1.Framework.Alignment.Near

' カレンダーの異なる日付に境界線を設定します
C1CalendarView1.Theme.Day.Bolded.Border = New C1.Framework.Thickness(2, 2, 2, 2)
C1CalendarView1.Theme.Day.Ordinary.Border = New C1.Framework.Thickness(1, 1, 1, 1)
C1CalendarView1.Theme.Day.Weekend.Border = New C1.Framework.Thickness(1, 1, 1, 1)

' カレンダーやカレンダー内の日付とタイトルの境界線の色を設定します
C1CalendarView1.Theme.Day.Bolded.BorderColor = Color.Crimson
C1CalendarView1.Theme.Day.Ordinary.BorderColor = Color.PeachPuff
C1CalendarView1.Theme.Day.Today.BorderColor = SystemColors.ControlLightLight
C1CalendarView1.Theme.Day.Weekend.BorderColor = Color.LightPink
C1CalendarView1.Theme.Titles.Day.BorderColor = Color.Crimson
C1CalendarView1.Theme.Titles.Week.BorderColor = Color.Crimson
C1CalendarView1.Theme.Titles.Weekend.BorderColor = Color.DarkCyan
C1CalendarView1.Theme.Common.BorderColor = Color.PeachPuff

' カレンダーの日付とタイトルにフォント設定を適用します
C1CalendarView1.Theme.Day.Bolded.Font =
    New Font("Microsoft Sans Serif", 8.25F,
             (FontStyle.Bold Or FontStyle.Underline))
C1CalendarView1.Theme.Day.Weekend.Font =
    New Font("Microsoft Sans Serif", 8.25F,
             FontStyle.Bold, GraphicsUnit.Point, 0)
C1CalendarView1.Theme.Titles.Day.Font =
    New Font("Microsoft Sans Serif", 8.25F,
             FontStyle.Bold, GraphicsUnit.Point, 0)
C1CalendarView1.Theme.Titles.Month.Font =
    New Font("Microsoft Sans Serif", 9.0F,
             (FontStyle.Bold Or FontStyle.Underline), GraphicsUnit.Point, 0)
C1CalendarView1.Theme.Titles.Week.Font =
    New Font("Microsoft Sans Serif", 6.0F,
             FontStyle.Bold)
```

```
' カレンダーの日付色およびタイトルの前景色を設定します
C1CalendarView1.Theme.Day.Bolded.ForeColor = Color.Crimson
C1CalendarView1.Theme.Day.Disabled.ForeColor = SystemColors.ControlText
C1CalendarView1.Theme.Day.Trail.ForeColor = Color.Transparent
C1CalendarView1.Theme.Day.Weekend.ForeColor = Color.Crimson
C1CalendarView1.Theme.Titles.Month.ForeColor = Color.Crimson

' カレンダーやカレンダー内の日付とタイトルの背景色を設定します
C1CalendarView1.Theme.Day.Bolded.BackColor = Color.Pink
C1CalendarView1.Theme.Day.Disabled.BackColor = Color.LightGray
C1CalendarView1.Theme.Day.Ordinary.BackColor = Color.White
C1CalendarView1.Theme.Day.Trail.BackColor = Color.White
C1CalendarView1.Theme.Titles.Day.BackColor = Color.PeachPuff
C1CalendarView1.Theme.Titles.Week.BackColor = Color.PeachPuff
C1CalendarView1.Theme.Day.Weekend.BackColor = Color.Pink
C1CalendarView1.Theme.Common.BackColor = Color.Linen
```

- **C#**

```
// タイトルおよびカレンダーの水平方向と垂直方向の配置を設定します
c1CalendarView1.Theme.Titles.Week.HorizontalAlignment = C1.Framework.Alignment.Far;
c1CalendarView1.Theme.Common.HorizontalAlignment = C1.Framework.Alignment.Near;
c1CalendarView1.Theme.Titles.Week.VerticalAlignment = C1.Framework.Alignment.Far;
c1CalendarView1.Theme.Common.VerticalAlignment = C1.Framework.Alignment.Near;

// カレンダーの異なる日付に境界線を設定します
c1CalendarView1.Theme.Day.Bolded.Border = new C1.Framework.Thickness(2, 2, 2, 2);
c1CalendarView1.Theme.Day.Ordinary.Border = new C1.Framework.Thickness(1, 1, 1, 1);
c1CalendarView1.Theme.Day.Weekend.Border = new C1.Framework.Thickness(1, 1, 1, 1);

// カレンダーやカレンダー内の日付とタイトルの境界線の色を設定します
c1CalendarView1.Theme.Day.Bolded.BorderColor = Color.Crimson;
c1CalendarView1.Theme.Day.Ordinary.BorderColor = Color.PeachPuff;
c1CalendarView1.Theme.Day.Today.BorderColor = SystemColors.ControlLightLight;
c1CalendarView1.Theme.Day.Weekend.BorderColor = Color.LightPink;
c1CalendarView1.Theme.Titles.Day.BorderColor = Color.Crimson;
c1CalendarView1.Theme.Titles.Week.BorderColor = Color.Crimson;
c1CalendarView1.Theme.Titles.Weekend.BorderColor = Color.DarkCyan;
c1CalendarView1.Theme.Common.BorderColor = Color.PeachPuff;

// カレンダーの日付とタイトルにフォント設定を適用します
c1CalendarView1.Theme.Day.Bolded.Font =
    new Font ("Microsoft Sans Serif", 8.25F,
    (FontStyle.Bold | FontStyle.Underline));
c1CalendarView1.Theme.Day.Weekend.Font =
    new Font("Microsoft Sans Serif", 8.25F,
    FontStyle.Bold, GraphicsUnit.Point, 0);
c1CalendarView1.Theme.Titles.Day.Font =
    new Font("Microsoft Sans Serif", 8.25F,
    FontStyle.Bold, GraphicsUnit.Point, 0);
c1CalendarView1.Theme.Titles.Month.Font =
    new Font("Microsoft Sans Serif", 9F,
    (FontStyle.Bold | FontStyle.Underline), GraphicsUnit.Point, 0);
c1CalendarView1.Theme.Titles.Week.Font =
    new Font("Microsoft Sans Serif", 6F, FontStyle.Bold);

// カレンダーの日付色およびタイトルの前景色を設定します
c1CalendarView1.Theme.Day.Bolded.ForeColor = Color.Crimson;
c1CalendarView1.Theme.Day.Disabled.ForeColor = SystemColors.ControlText;
c1CalendarView1.Theme.Day.Trail.ForeColor = Color.Transparent;
c1CalendarView1.Theme.Day.Weekend.ForeColor = Color.Crimson;
c1CalendarView1.Theme.Titles.Month.ForeColor = Color.Crimson;
```

```
// カレンダーやカレンダー内の日付とタイトルの背景色を設定します
c1CalendarView1.Theme.Day.Bolded.BackColor = Color.Pink;
c1CalendarView1.Theme.Day.Disabled.BackColor = Color.LightGray;
c1CalendarView1.Theme.Day.Ordinary.BackColor = Color.White;
c1CalendarView1.Theme.Day.Trail.BackColor = Color.White;
c1CalendarView1.Theme.Titles.Day.BackColor = Color.PeachPuff;
c1CalendarView1.Theme.Titles.Week.BackColor = Color.PeachPuff;
c1CalendarView1.Theme.Day.Weekend.BackColor = Color.Pink;
c1CalendarView1.Theme.Common.BackColor = Color.Linen;
```

DateEditの機能

DateEdit では、選択または入力した日付を操作するために役立つさまざまな機能が提供されています。

日付書式
: DateEdit で、定義済みの書式またはカスタム書式で日付を表示する方法を学習します。

null 値とウォーターマークのサポート
: DateEdit で、null 値を処理する方法およびウォーターマークを表示する方法を学習します。

キーボードサポート
: DateEdit でキーの組み合わせを使用する方法を学習します。

マスク
: DateEdit でマスクによってユーザーの入力を制限する方法を学習します。

データ検証
: DateEdit で事前検証と事後検証を実装する方法を学習します。

値の書式設定と解析
: DateEdit で値の書式設定および解析を行う方法を学習します。

国際化
: DateEdit で、現在のカルチャ設定を変更する方法および右から左のサポートを実装する方法を学習します。

外観とスタイル設定
: DateEdit の外観をカスタマイズする方法を学習します。

日付書式

DateEdit コントロールでは、多くの日付書式がサポートされていますが、これらは次のように分類できます。

- **定義済み書式**: DateEdit では、短い日付、長い日付、標準の日付、短い時刻、長い時刻、指数、パーセントなど、定義済みの書式がサポートされています。これらの書式のいずれかで日時を表示するには、**FormatType** プロパティの値を **FormatTypeEnum** のいずれかの値に設定します。
- **カスタム書式**: DateEdit では、日付のカスタム書式設定が可能です。カスタム書式を指定して、その書式で日付を表示できます。カスタム書式設定を有効にするには、**FormatType** プロパティを FormatTypeEnum の **CustomFormat** に設定します。さらに、カスタム書式で日付を表示するには、**CustomFormat** プロパティでカスタム書式指定子を設定します。

カスタム書式指定子については、「カスタム日時書式文字列」を参照してください。DateEdit コントロールは、ここに記載されている書式文字列の大部分をサポートします。

次の図に、カスタム書式で日付が表示されている DateEdit コントロールを示します。

次のコードスニペットは、DateEdit で日付をカスタム書式で表示する方法を示します。

- **Visual Basic**

```
' 形式をカスタムとして指定します
C1DateEdit1.FormatType = C1.Win.C1Input.FormatTypeEnum.CustomFormat

' カスタム書式の指示子を設定します
C1DateEdit1.CustomFormat = "MMMM-dd-yyyy"
```

- **C#**

```
// 形式設定をカスタムフォーマットとして指定します
c1DateEdit1.FormatType = C1.Win.C1Input.FormatTypeEnum.CustomFormat;

// カスタム書式の指示子を設定します
c1DateEdit1.CustomFormat = "MMMM-dd-yyyy";
```

null 値とウォーターマークのサポート

null 値の処理は、適切な規則や設定がないと難しい場合があります。しかし、DateEdit では、さまざまなケースで null 値を処理し、関連する要素を制御するための柔軟な規則が提供されています。

読み取り専用モードの場合

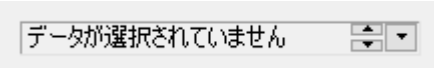
DateEdit コントロールでは、コントロールが読み取り専用モードの場合に、**DisplayFormat** の **NullText** プロパティを使用して null 値を表示できます。DisplayFormat で NullText プロパティをオーバーライドして、DateEdit コントロールにウォーターマークを表示することができます。

編集モードの場合

これは DateEditor のデフォルトのモードです。編集モードでは（ReadOnly が false の場合）、DateEdit は、**EditFormat** の **NullText** プロパティを設定することで、null 値を表示できます。

どちらのモードでも、**EmptyAsNull** プロパティを **true** に設定することで、空の文字列を表示することができます。**DateTimeInput** が **true** に設定されている DateEdit で日時の値を編集するときは、空のコントロールによって null 値が表されます。コントロールで値の編集を開始すると、コントロールは最後に割り当てられた null 以外の値または今日の日付に変わります。

次の図に、読み取り専用モードの DateEdit に表示されたヌルテキスト（ウォーターマーク）を示します。



次のコードスニペットは、読み取り専用モードの DateEdit でヌルテキストを表示する方法を示します。

- **Visual Basic**

```
' カレンダーポップアップを読み取り専用に設定します
C1DateEdit1.[ReadOnly] = True

' DateEditにnullテキストを表示します
C1DateEdit1.DisplayFormat.NullText = "データが選択されていません"
```

- **C#**

```
// カレンダーポップアップを読み取り専用に設定します
c1DateEdit1.ReadOnly = true;

// DateEditにnullテキストを表示します
c1DateEdit1.DisplayFormat.NullText = "データが選択されていません";
```

キーボードサポート

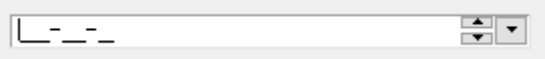
次の表に、DateEdit コントロールで適用できるキーの組み合わせと対応するアクションを示します。

キーボードコマンド	アクション
↑	選択されている値を 1 だけ増加させます。
↓	選択されている値を 1 だけ減少させます。
←	選択を 1 つ左の位置に移動します。
→	選択を 1 つ右の位置に移動します。
Home	選択をコンボボックス内の最初の値に移動します。
End	選択をコンボボックス内の最後の値に移動します。
Shift + ↑/←	現在の選択から 1 つ左の位置の値まで連続して値を選択します。
Shift + ↓/→	現在の選択から 1 つ右の位置の値まで連続して値を選択します。
Shift + Home	現在の選択からコンボボックス内の最初の値までのすべての値を選択します。
Shift + End	現在の選択からコンボボックス内の最後の値までのすべての値を選択します。
Ctrl + ↑/←	同じグループの左方向にある値をカーソルがスキップします。
Ctrl + ↓/→	同じグループの右方向にある値をカーソルがスキップします。
Ctrl + Home	現在の選択から最初の日にカーソルを移動します。
Ctrl + End	現在の選択から最後の日にカーソルを移動します。

マスク

マスクにより、エディタでのユーザー入力を制限することができます。DateEdit コントロールで入力をマスクすると、値を特定の書式で入力することができます。たとえば、DateEdit コントロールが数値だけを受け取るようにしたり、24 時間形式の日時の値だけを受け取るようにすることができます。ユーザーがこのような書式で値を入力するように制限するには、DateEdit コントロールでマスクを使用します。

次の図に、編集モードでマスクが適用された DateEdit コントロールを示します。



DateEdit では、**EditMask** プロパティを使用して入力をマスクできます。マスク入力を有効にするには、**EditMask** プロパティに、プレースホルダとリテラルから成るマスク文字列を設定します。**MaskInfo** プロパティを使用して、マスク入力の他の要素を制御できます。また、**CustomPlaceholders** プロパティを使用して、ユーザー定義のマスク文字で構成される独自のプレースホルダを定義することもできます。DateEdit コントロールが読み取り専用モードの場合、マスクは機能しません。

次の表に、DateEdit でサポートされているすべてのプレースホルダおよびリテラルを示します。

プレースホルダ	説明
#	桁のプレースホルダを指定すると、その位置に英数字またはプラス/マイナス記号を入力できます（入力はオプション）。
.	小数点プレースホルダ。実際に使用される文字は、国際化設定で小数点のプレースホルダとして指定された文字です。この文字は、リテラルとしてマスク処理されます。

,	千単位の桁区切りです。実際に使用される文字は、国際化設定で桁区切りとして指定された文字です。この文字は、リテラルとしてマスク処理されます。
:	時刻区切り文字。実際に使用される文字は、国際化設定で時刻区切りとして指定された文字です。この文字は、リテラルとしてマスク処理されます。
/	日付区切り文字。実際に使用される文字は、国際化設定で日付区切りとして指定された文字です。この文字は、リテラルとしてマスク処理されます。
\	マスク文字列内の次の文字をリテラルとして処理します。これを使用して、#、&、A、... などの文字をマスクに入れることができます。この文字は、リテラルとしてマスク処理されます。
&	文字のプレースホルダです（入力は必須）。任意の文字を入力できます。
>	後続のすべての文字を大文字に変換します。
<	後続のすべての文字を小文字に変換します。
~	前出の < または > を無効にします。
!	編集マスク内の後続のオプションの文字が左から右ではなく、右から左に表示されます。このため、空白は左に表示されます。
^	前出の！文字を無効にします。^ の後では、空白は右に表示されます。
A	英数字のプレースホルダです（入力は必須）。たとえば、a – z、A – Z、0 – 9 です。
a	英数字のプレースホルダです（入力はオプション）。
0	桁のプレースホルダです（入力は必須）。たとえば、0 – 9 です。
9	桁のプレースホルダです（入力はオプション）。
C	文字またはスペースのプレースホルダです（入力はオプション）。任意の文字を入力できます。
L	英文字のプレースホルダです（入力は必須）。たとえば、a – z、A – Z です。
?	英文字のプレースホルダです（入力はオプション）。
\n	改行を示すリテラルです。**Multiline** プロパティが **True** に設定されている場合に適用できます。
"	二重引用符で囲まれた文字列内のすべての文字はリテラルと見なされます。
リテラル	その他のすべての記号はリテラルとして、つまりそのまま表示されます。

次のコードスニペットは、EditMask プロパティを設定して DateEdit でマスクを適用する方法を示します。

- **Visual Basic**

```
' EditMaskプロパティを設定してマスクを適用します
C1DateEdit1.EditMask = "9900-LLL-00"
```

- **C#**

```
// EditMaskプロパティを設定してマスクを適用します
c1DateEdit1.EditMask = "9900-LLL-00";
```

データ検証

DateEdit コントロールでは、事前検証と事後検証という 2 種類のデータ検証がサポートされています。事前検証は、入力文字

列自体のデータ検証であるのに対して、事後検証は、ユーザーによって入力された型付きの値のデータ検証です。

以下の各セクションで、DateEdit における事前検証と事後検証について説明します。

事前検証
DateEdit で事前検証を実装する方法を学習します。

事後検証
DateEdit で事後検証を実装する方法を学習します。

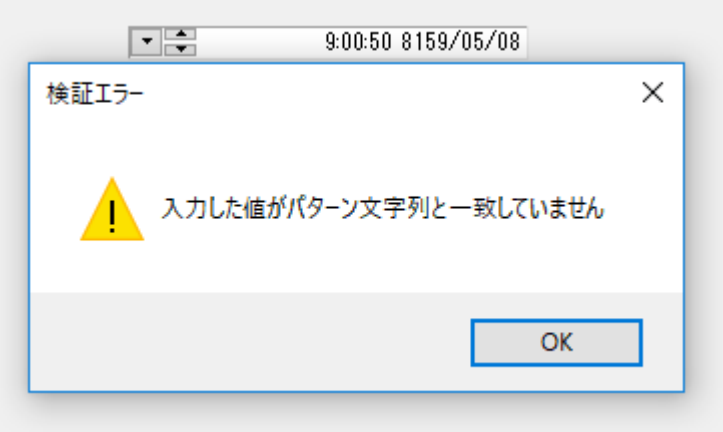
事前検証

事前検証は、入力された文字列自体を検証します。つまり、解析する前の文字列に規則が適用されます。

DateEdit では、**PreValidation** プロパティを使用して入力文字列の事前検証を制御できます。コントロールには、次に示すように、さまざまな事前検証オプションが用意されています。

- **大文字小文字を区別した比較**: 文字列比較で大文字小文字を区別できます。大文字小文字を区別した比較を有効にするには、**CaseSensitive** プロパティを **true** に設定します。
- **検証パターン**: 検証パターン（規則）を含む文字列を指定できます。検証パターンを指定するには、**PatternString** プロパティを設定します。PatternString で項目セパレータを使用することで、複数の規則や部分文字列を指定できます。DateEdit では、複数の文字列の項目セパレータとして、デフォルトで「|」がサポートされています。ただし、**ItemSeparator** プロパティを設定することで、項目セパレータを変更できます。
- **検証方法**: 検証方法を指定できます。検証方法を指定するには、**Validation** プロパティを **PreValidationTypeEnum** に含まれる次の値のいずれかに設定します。
  - **ExactList**: PatternString には、有効な値が ItemSeparator で区切って指定されます。
  - **PreValidatingEvent**: 検証には PreValidating イベントが使用されます。
  - **Wildcards**: PatternString には、ワイルドカードパターンが ItemSeparator で区切って指定されます。パターン内では、?（任意の 1 文字）、#（任意の 1 桁）、*（0 個以上の文字）、\（エスケープ）の各文字が予約されています。また、PreValidation プロパティを使用して、カスタムパターン文字を定義することもできます。
  - **RegexPattern**: PatternString には、正規表現が指定されます。
- **エラーメッセージ**: 入力文字列が検証パターンと一致しない場合に表示されるメッセージを設定できます。エラーメッセージを設定するには、**ErrorMessage** プロパティを設定します。

次の図に、検証エラーメッセージが表示された DateEdit コントロールを示します。



次のコードスニペットは、さまざまなプロパティを設定して、DateEdit で事前検証を適用する方法を示します。

- **Visual Basic**

```
' DateEditにPreValidationを適用します
C1DateEdit1.PreValidation.CaseSensitive = True
C1DateEdit1.PreValidation.ErrorMessage = "入力した値がパターン文字列と一致していません"
C1DateEdit1.PreValidation.ItemSeparator = "|"
C1DateEdit1.PreValidation.PatternString = "00|11"
```

```
C1DateEdit1.PreValidation.Validation = C1.Win.C1Input.PreValidationTypeEnum.ExactList
```

- **C#**

```
// DateEditにPreValidationを適用します
c1DateEdit1.PreValidation.CaseSensitive = true;
c1DateEdit1.PreValidation.ErrorMessage = "入力した値がパターン文字列と一致していません";
c1DateEdit1.PreValidation.ItemSeparator = "|";
c1DateEdit1.PreValidation.PatternString = "00|11";
c1DateEdit1.PreValidation.Validation = C1.Win.C1Input.PreValidationTypeEnum.ExactList;
```

事後検証

事後検証は、解析後に適用される検証規則です。ユーザーによって入力された値を検証できます。

DateEdit の事後検証は、**PostValidation** プロパティによって制御されます。コントロールには、次に示すように、さまざまな事後検証オプションが用意されています。

- **定義済みの値**: 入力値が定義済みの値リスト内の値と一致するかどうかをチェックできます。定義済みの値を指定するには、**Values** プロパティを設定します。
- **除外される値**: 入力値として許可されない値を指定できます。これらの値を設定するには、**ValuesExcluded** プロパティを設定します。
- **範囲**: 入力値が特定の範囲に含まれるかどうかをチェックできます。DateEdit では複数の範囲もサポートされ、**Intervals** プロパティを使用してそれらを指定できます。
- **検証方法**: 検証方法を指定できます。**Validation** プロパティを、宣言型の検証の場合は **ValuesAndIntervals**、プログラムによる検証の場合は **PostValidatingEvent** にそれぞれ設定します。

次のコードスニペットは、さまざまなプロパティを設定して、DateEdit で事後検証を適用する方法を示します。

- **Visual Basic**

```
' DateEditにPostValidationを適用します
C1DateEdit1.PostValidation.Validation = C1.Win.C1Input.PostValidationTypeEnum.ValuesAndIntervals
C1DateEdit1.PostValidation.Values = New ValueType() {1, 2, 4}
C1DateEdit1.PostValidation.ErrorMessage = "Wrong"
C1DateEdit1.PostValidation.CaseSensitive = False
```

- **C#**

```
// DateEditにPostValidationを適用します
c1DateEdit1.PostValidation.Validation = C1.Win.C1Input.PostValidationTypeEnum.ValuesAndIntervals;
c1DateEdit1.PostValidation.Values = new[] { 1, 2, 4 };
c1DateEdit1.PostValidation.ErrorMessage = "Wrong";
c1DateEdit1.PostValidation.CaseSensitive = false;
```

値の書式設定と解析

書式設定

DateEdit コントロールでは、**FormatType** プロパティを使用して値の書式設定を制御できます。このプロパティの列挙値は、コントロール内でのデータの書式設定方法を定義します。数値型と日時型の一部のオプションは、.NET 標準の書式指定子に対応します。たとえば、StandardNumber や LongDate です。DateEdit でサポートされている日付書式については、「日付書式」を参照してください。

定義済みの書式のほかに、**CustomFormat** プロパティで指定されているカスタム書式指定子に対応するカスタム書式があります。また、UseEvent という特別な書式タイプもあります。これを使用すると、書式設定が Formatting イベントによって決定されます。さらに、NULL 値（System.DBNull）を表す方法は、**NullText** プロパティと **EmptyAsNull** プロパティによって制御されます。DateEdit での null 値の処理については、「null 値とウォーターマークのサポート」を参照してください。

書式設定された値を表示する際に、先頭または末尾にあるスペースを削除したい場合があります。先頭または末尾のスペースを削除するには、それぞれ **TrimStart** プロパティと **TrimEnd** プロパティを設定します。

DateEdit では、表示用（コントロールが読み取り専用であるとき、つまり編集モードでない場合）と編集モード用の 2 つの書式がサポートされています。これらの書式設定モードには、**DisplayFormat** プロパティと **EditFormat** プロパティを使用してアク

セスできます。

解析

ユーザーによって入力された文字列をデータ型に変換することを解析といいます。これは、書式設定とは逆の処理です。DateEdit コントロールでは、**ParseInfo** プロパティを使用して解析を制御できます。ParseInfo プロパティから **ParseInfo** クラスにアクセスできます。このクラスには、解析のさまざまな側面を制御するサブプロパティがあります。

デフォルトの解析は大半のケースに十分対応できますが、コントロールによる解析方法のさまざまな要素を変更することもできます。それには、ParseInfo プロパティを拡張し、(Inherit) フラグを変更して、必要なプロパティを設定します。デフォルトでは、書式設定に使用される書式設定プロパティ値が解析にも使用されます。

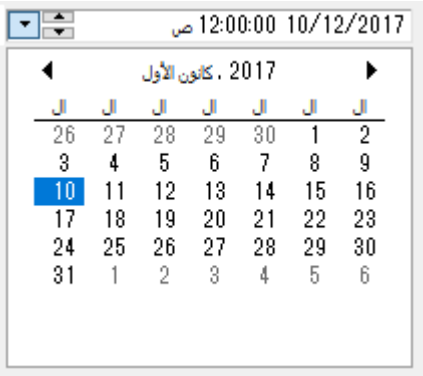
国際化

DateEdit では、次の方法で国際化がサポートされています。

- **カルチャ設定**: DateEdit では、特定のカルチャに基づいたデータの書式設定、解析、検証が可能です。このコントロールでは、System.Globalization 名前空間の CultureInfo クラスを使用してカルチャを指定できます。DateEdit のカルチャを指定するには、**CurrentCulture** プロパティを設定します。さまざまなカルチャ名と識別子については、.NET Framework ドキュメントの CultureInfo クラスを参照してください。
- **右から左のサポート**: DateEdit は、右から左への筆記法に従う言語向けに右から左のサポートを提供しています。DateEdit コントロールで右から左の機能を有効にするには、Systems.Windows.Forms の RightToLeft コントロールの **RightToLeft** プロパティを設定します。

CalendarView でカルチャ設定と右から左の機能を設定する方法については、「CalendarView の機能」の「カルチャ設定」と「右から左のサポート」を参照してください。

次の図に、アラビア語カルチャが適用され、右から左の方向で表示された DateEdit を示します。



次のコードスニペットは、DateEdit コントロールで現在のカルチャを設定し、右から左の機能を有効にする方法を示します。

- **Visual Basic**

```
' 現在のカルチャを設定します
C1DateEdit1.Culture = New System.Globalization.CultureInfo("ar-IQ").LCID

' 右から左への機能を有効にします
C1DateEdit1.RightToLeft = RightToLeft.Yes
```

- **C#**

```
// 現在のカルチャを設定します
c1DateEdit1.Culture = new System.Globalization.CultureInfo("ar-IQ").LCID;

// 右から左への機能を有効にします
c1DateEdit1.RightToLeft = RightToLeft.Yes;
```

外観とスタイル設定

DateEdit コントロールの外観は、コントロールを使用するアプリケーションに適合するようにカスタマイズできます。DateEdit には以下のスタイル設定機能が用意されており、背景色や前景色の変更、境界線の追加、テキスト配置の変更、テーマの設定などを行うことができます。

- **よく使用されるスタイル**:境界線の色、境界線スタイル、フォント設定、背景色、前景色など、よく使用されるスタイルを DateEdit コントロールに適用できます。
  - **境界線の色**:境界線の色を変更するには、**BorderColor** プロパティを設定します。
  - **境界線スタイル**:コントロールの境界線スタイルを設定するには、**BorderStyle** プロパティを設定します。
  - **フォント設定**:DateEdit コントロール内のテキストにフォント設定を適用するには、Systems.Windows.Forms の Font コントロールの **Font** プロパティを設定します。
  - **背景色**:DateEdit の背景色を変更するには、**BackColor** プロパティを設定します。
  - **前景色**:DateEdit に前景色を適用するには、**ForeColor** プロパティを設定します。
- **テキスト配置**:Systems.Windows.Forms の TextBox の **TextAlign** プロパティを設定して、DateEdit コントロール内のテキストの配置を設定できます。
- **テーマ**:**VisualStyle** プロパティを設定することで、さまざまなテーマを適用して DateEdit の全体的な外観を強化できます。

カレンダーポップアップまたは CalendarView の外観のカスタマイズについては、「CalendarView の機能」の「外観とスタイル設定」を参照してください。

次の図は、DateEdit コントロールにスタイル設定機能を適用したところです。

次のコードスニペットは、関連するプロパティを使用して CalendarView でスタイル設定機能を実装する方法を示します。

- **Visual Basic**

```
' 境界線の色を設定します
C1DateEdit1.BorderColor = Color.DarkBlue

' 境界線スタイルを設定します
C1DateEdit1.BorderStyle = BorderStyle.FixedSingle

' フォント設定を適用します
C1DateEdit1.Font = New Font("Microsoft Sans Serif", 9.0F, FontStyle.Italic)

' 前景色と背景色を設定します
C1DateEdit1.ForeColor = Color.Brown
C1DateEdit1.BackColor = Color.LightCyan

' テキストの配置を設定します
C1DateEdit1.TextAlign = HorizontalAlignment.Left
```

- **C#**

```
// 境界線の色を設定します
c1DateEdit1.BorderColor = Color.DarkBlue;

// 境界線スタイルを設定します
c1DateEdit1.BorderStyle = BorderStyle.FixedSingle;

// フォント設定を適用します
c1DateEdit1.Font = new Font("Microsoft Sans Serif", 9F, FontStyle.Italic);

// 前景色と背景色を設定します
c1DateEdit1.ForeColor = Color.Brown;
c1DateEdit1.BackColor = Color.LightCyan;
```

```
// テキストの配置を設定します
c1DateEdit1.TextAlign = HorizontalAlignment.Left;
```