Calendar for UWP

2018.03.06 更新

グレープシティ株式会社

Calendar for UWP

目次

Calendar for UWP

Calendar for UWP を使用して、日付中心のダッシュボードやスケジュールアプリケーションを作成します。C1Calendar コントロールは、日付のナビゲーションや日付範囲の選択に使用されます。予定などのカスタムカレンダー情報を表示することもできます。単一の月を表示したり、任意の日数を選択するために使用できます。単純な書式設定やカスタムコンテンツで日付を強調表示することができます。

次のトピックに従って、**Calendar for UWP** を簡単に使用できます。

主な特長

Calendar 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。**Calendar for UWP** は、次の主要な機能を備えています。

- **ジェスチャベースの月のナビゲーション**
  C1Calendar は、スライドジェスチャとフリックジェスチャによる月のナビゲーションをデフォルトでサポートします。ナビゲーションボタンをタップしても同様の操作を実行できます。
- **日付範囲選択**
  通常のタップや押してスライドのジェスチャにより、1日や複数の日を選択できます。選択できる日数は、**MaxSelectionCount** プロパティを使用して制御します。
- **カスタマイズ可能なカレンダー設定**
  週の最初の曜日、営業日などを指定することができます。UWP でサポートされているカルチャを設定して **C1Calendar** を使用できます。営業日と週末の休日を特別なブラシプロパティで視覚的に区別できます。任意の日付を太字にして、エンドユーザーに強調表示することができます。
- **簡単で柔軟なスタイル設定モデル**
  C1Calendar では、テンプレートを上書きしなくてもコントロールのブラシを簡単に変更できます。前後の日、週末、選択中の日、月ヘッダーなど、コントロールの各表示部分は独自のブラシを持ちます。
- **日付の外観とコンテンツのカスタマイズ**
  カスタムテンプレートとテンプレートセレクタを使用して、個々の日の外観を変更することができます。**C1DataTemplateSelector** の独自の実装を使用して、日付ブロック内にカレンダー予定などのカスタムコンテンツを表示できます。
- **週番号の表示**
  1つの単純なプロパティを設定するだけで、週番号を表示できます。**C1Calendar** は、1か月が4週でも5週でも6週でも、常に必要な行数のみを表示します。末尾に空の週は表示されません。
- **年カレンダー および 10年カレンダーモード**
  年カレンダーモードまたは10年カレンダーモードを使用して歴史的日付（数年前の日付）を簡単に入力できます。月ヘッダーをタップして年カレンダーモードに、そして年ヘッダーをタップして10年カレンダーモードに変更できます。詳細については、「手順3：Calendar アプリケーションの実行」をご参照ください。

クイックスタート

このクイックスタートガイドは、**Calendar 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では、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 **Calendar for UWP** コントロールを追加して、コントロールの外観と動作をカスタマイズします。

手順1：C1Calendar 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、Visual Studio で、**Calendar for UWP** を使用して UWP 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 で、次の `<Page.Resources>` マークアップを `<Page>` タグと `<Grid>` タグの間に追加して、コントロールをカスタマイズします。

XAML でマークアップの書き方

XAML マークアップ

```
<Page.Resources>
    <!-- カレンダー日のカスタムDataTemplatesを返します。-->
    <local:DaySlotTemplateSelector x:Key="DaySlotTemplateSelector">
        <Calendar:DaySlotTemplateSelector.Resources>
            <ResourceDictionary>
                <DataTemplate x:Key="BoldedDay">
                    <Grid>
                        <Grid.RowDefinitions>
                            <RowDefinition />
                            <RowDefinition Height="Auto"/>
                        </Grid.RowDefinitions>
                        <!-- DaySlot.Tag プロパティに保存した予定情報を表示します。 -->
                        <Border Background="LightGreen" Grid.Row="0"
VerticalAlignment="Top" >
                            <TextBlock Text="{Binding Tag}" Margin="5"
TextWrapping="Wrap" Foreground="Black" />
                        </Border>
                        <TextBlock Text="{Binding}" Grid.Row="1"
Foreground="OrangeRed" HorizontalAlignment="Left" VerticalAlignment="Bottom"
FontWeight="SemiBold" Margin="6,0,0,4"/>
                    </Grid>
                </DataTemplate>
                <DataTemplate x:Key="UnboldedDay">
                    <TextBlock Text="{Binding}" HorizontalAlignment="Left"
VerticalAlignment="Bottom" Margin="6,22,0,4"/>
                </DataTemplate>
            </ResourceDictionary>
        </Calendar:DaySlotTemplateSelector.Resources>
    </local:DaySlotTemplateSelector>
    <local:SmallDaySlotTemplateSelector x:Key="SmallDaySlotTemplateSelector">
```

```
        <Calendar:DaySlotTemplateSelector.Resources>
            <ResourceDictionary>
                <DataTemplate x:Key="BoldedDay">
                    <TextBlock Text="{Binding}" HorizontalAlignment="Left"
VerticalAlignment="Bottom" FontWeight="ExtraBlack" Margin="10,8,5,8"/>
                </DataTemplate>
                <DataTemplate x:Key="UnboldedDay">
                    <TextBlock Text="{Binding}" HorizontalAlignment="Left"
VerticalAlignment="Bottom" Margin="6,12,5,4"/>
                </DataTemplate>
            </ResourceDictionary>
        </Calendar:DaySlotTemplateSelector.Resources>
    </local:SmallDaySlotTemplateSelector>
</Page.Resources>
```

4. ツールボックスに移動し、C1Calendar 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。これで、参照と XAML 名前空間が自動的に追加されます。XAML マークアップは次のようになります。

XAML マークアップ

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
    <Calendar:C1Calendar/>
</Grid>
```

5. 次のマークアップを `<Calendar:C1Calendar>` タグに追加して、コントロールをカスタマイズします。

XAML マークアップ

```
<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">
    <Calendar:C1Calendar x:Name="Calendar" Margin="20" Grid.Row="0"
SelectedDateChanged="Calendar_SelectedDateChanged" DayOfWeekFormat="dddd"
MaxSelectionCount="21" ShowWeekNumbers="true" WeekendBrush="Red"/>
</Grid>
```

これは、コントロールに名前を付け、カレンダーの書式設定と外観をカスタマイズします。後の手順で、参照されるイベントハンドラのコードを追加します。

6. 次のマークアップを Calendar の上の `<Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}">` タグと `<Calendar:C1Calendar>` タグの間に追加します。

XAML でマークアップの書き方

XAML マークアップ

```
<!--アプリのオリエンテーション状態-->
        <VisualStateManager.VisualStateGroups>
            <VisualStateGroup x:Name="OrientationStates">
                <VisualState x:Name="Full"/>
                <VisualState x:Name="Fill">
                    <Storyboard>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(FrameworkElement.Margin)"
Storyboard.TargetName="Calendar">
                            <DiscreteObjectKeyFrame KeyTime="0">
                                <DiscreteObjectKeyFrame.Value>
                                    <Thickness>15</Thickness>
                                </DiscreteObjectKeyFrame.Value>
                            </DiscreteObjectKeyFrame>
```

```
                        </ObjectAnimationUsingKeyFrames>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(C1Calendar.DayOfWeekFormat)"
Storyboard.TargetName="Calendar">
                            <DiscreteObjectKeyFrame KeyTime="0" Value="ddd"/>
                        </ObjectAnimationUsingKeyFrames>
                    </Storyboard>
                </VisualState>
                <VisualState x:Name="Portrait">
                    <Storyboard>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(FrameworkElement.Margin)"
Storyboard.TargetName="Calendar">
                            <DiscreteObjectKeyFrame KeyTime="0">
                                <DiscreteObjectKeyFrame.Value>
                                    <Thickness>15</Thickness>
                                </DiscreteObjectKeyFrame.Value>
                            </DiscreteObjectKeyFrame>
                        </ObjectAnimationUsingKeyFrames>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(C1Calendar.DayOfWeekFormat)"
Storyboard.TargetName="Calendar">
                            <DiscreteObjectKeyFrame KeyTime="0" Value="ddd"/>
                        </ObjectAnimationUsingKeyFrames>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(C1Calendar.ShowWeekNumbers)"
Storyboard.TargetName="Calendar">
                            <DiscreteObjectKeyFrame KeyTime="0" Value="false"/>
                        </ObjectAnimationUsingKeyFrames>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(FrameworkElement.Visibility)"
Storyboard.TargetName="SelectedDayInfo">
                            <DiscreteObjectKeyFrame KeyTime="0" >
                                <DiscreteObjectKeyFrame.Value>
                                    <Visibility>Visible</Visibility>
                                </DiscreteObjectKeyFrame.Value>
                            </DiscreteObjectKeyFrame>
                        </ObjectAnimationUsingKeyFrames>
                    </Storyboard>
                </VisualState>
                <VisualState x:Name="Snapped">
                    <Storyboard>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(FrameworkElement.Margin)"
Storyboard.TargetName="Calendar">
                            <DiscreteObjectKeyFrame KeyTime="0">
                                <DiscreteObjectKeyFrame.Value>
                                    <Thickness>5</Thickness>
                                </DiscreteObjectKeyFrame.Value>
                            </DiscreteObjectKeyFrame>
                        </ObjectAnimationUsingKeyFrames>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(C1Calendar.ShowWeekNumbers)"
```

```
Storyboard.TargetName="Calendar">
                            <DiscreteObjectKeyFrame KeyTime="0" Value="false"/>
                        </ObjectAnimationUsingKeyFrames>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(C1Calendar.DayOfWeekFormat)"
Storyboard.TargetName="Calendar">
                            <DiscreteObjectKeyFrame KeyTime="0" Value="d"/>
                        </ObjectAnimationUsingKeyFrames>
                        <ObjectAnimationUsingKeyFrames
Storyboard.TargetProperty="(FrameworkElement.Visibility)"
Storyboard.TargetName="SelectedDayInfo">
                            <DiscreteObjectKeyFrame KeyTime="0" >
                                <DiscreteObjectKeyFrame.Value>
                                    <Visibility>Visible</Visibility>
                                </DiscreteObjectKeyFrame.Value>
                            </DiscreteObjectKeyFrame>
                        </ObjectAnimationUsingKeyFrames>
                    </Storyboard>
                </VisualState>
            </VisualStateGroup>
        </VisualStateManager.VisualStateGroups>
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>
```

7. 次のマークアップを Calendar の下の `<Calendar:C1Calendar>` タグと `</Grid>` タグの間に追加します。

XAML でマークアップの書き方

XAML マークアップ

```
<Grid x:Name="SelectedDayInfo" Grid.Row="1" Height="120" Visibility="Collapsed"
>
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition/>
            </Grid.RowDefinitions>
            <StackPanel Orientation="Horizontal" Margin="10" Grid.Row="0">
                <TextBlock Text="SelectedDate: "/>
                <TextBlock Text="{Binding SelectedDate, ElementName=Calendar}"/>
            </StackPanel>
            <TextBlock x:Name="dayInfo" Margin="10" Grid.Row="1"
Foreground="Red"/>
</Grid>
```

ここまでの成果

これで、**C1Calendar** 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2：Calendar へのデータの追加」では、**C1Calendar** にデータを追加します。

手順2：Calendar へのデータの追加

前の手順では、**C1Calendar** コントロールをアプリケーションに追加しました。この手順では、**DataSeries** オブジェクトとそのデータを追加します。

プログラムでカレンダーにデータを追加するには、次の手順に従います。

1. **[表示]→[コード]**を選択してコードビューに切り替えます。
2. 次の imports 文をページの先頭に追加します。

Visual Basic コードの書き方

```
Visual Basic
Imports Windows.UI.ViewManagement
Imports C1.Xaml
Imports C1.Xaml.Calendar
```

C# コードの書き方

```
C#
using Windows.UI.ViewManagement;
using C1.Xaml;
using C1.Xaml.Calendar;
```

3. **MainPage** クラス内に次のコードを追加します。

Visual Basic コードの書き方

```
Visual Basic
' 予定の辞書
Private _boldedDays As New Dictionary(Of DateTime, String)()
Private _dayTemplateSelector As DaySlotTemplateSelector = Nothing
Private _loaded As Boolean = False
```

C# コードの書き方

```
C#
// 予定の辞書
private Dictionary<DateTime, string> _boldedDays = new Dictionary<DateTime,
string>();
private DaySlotTemplateSelector _dayTemplateSelector = null;
private bool _loaded = false;
```

4. **MainPage** コードを更新します。次のようになります。

Visual Basic コードの書き方

```
Visual Basic
Public Sub New()
    Me.InitializeComponent()
    AddHandler Window.Current.SizeChanged, AddressOf Current_SizeChanged
    Calendar.DayOfWeekSlotTemplateSelector = New DayOfWeekTemplateSelector()
    ' 太字の日を追加します
    _boldedDays.Add(DateTime.Today.AddDays(2), "スポーツ\r\nを忘れない。")
    _boldedDays.Add(DateTime.Today.AddDays(13), "誕生日")
    _boldedDays.Add(DateTime.Today.AddDays(22), "重要な会議")
```

```
    _boldedDays.Add(DateTime.Today.AddDays(-1), "8時に記念& vbCr & vbLf &パーティー")
    _boldedDays.Add(DateTime.Today.AddDays(-12), "医者との約束")
    _boldedDays.Add(DateTime.Today.AddDays(-21), "会議2日目")
    _boldedDays.Add(DateTime.Today.AddDays(-22), "会議1日目")
    For Each val As DateTime In _boldedDays.Keys
        Calendar.BoldedDates.Add(val)
    Next
End Sub
```

C# コードの書き方

C#

```
public MainPage()
{
    this.InitializeComponent();
    Window.Current.SizeChanged += Current_SizeChanged;
    Calendar.DayOfWeekSlotTemplateSelector = new DayOfWeekTemplateSelector();
    // 太字の日を追加します
    _boldedDays.Add(DateTime.Today.AddDays(2), "スポーツ\r\nを忘れない。");
    _boldedDays.Add(DateTime.Today.AddDays(13), "誕生日");
    _boldedDays.Add(DateTime.Today.AddDays(22), "重要な会議");
    _boldedDays.Add(DateTime.Today.AddDays(-1), "8時に記念\r\nパーティー");
    _boldedDays.Add(DateTime.Today.AddDays(-12), "医者との約束");
    _boldedDays.Add(DateTime.Today.AddDays(-21), "会議2日目");
    _boldedDays.Add(DateTime.Today.AddDays(-22), "会議1日目");
    foreach (DateTime val in _boldedDays.Keys)
    {
        Calendar.BoldedDates.Add(val);
    }
}
```

5. **MainPage** クラス内で、追加したコードの直後に次のコードを追加します。

Visual Basic コードの書き方

Visual Basic

```
Private Sub Calendar_SelectedDateChanged(sender As Object, e As
DateChangedEventArgs)
        If Calendar.SelectedDates.Count > 0 AndAlso
_boldedDays.ContainsKey(Calendar.SelectedDates(0)) Then
            dayInfo.Text = _boldedDays(Calendar.SelectedDates(0))
        Else
            dayInfo.Text = ""
        End If
    End Sub
   ''' <summary>
   ''' 日付を太字にするためのカスタム DataTemplate を使用する DayTemplateSelector
   ''' </summary>
    Private ReadOnly Property DayTemplateSelector() As DaySlotTemplateSelector
        Get
            If _dayTemplateSelector Is Nothing Then
                _dayTemplateSelector =
```

```
TryCast(Resources("DaySlotTemplateSelector"), DaySlotTemplateSelector)
                If _dayTemplateSelector IsNot Nothing Then
                    _dayTemplateSelector.BoldedDays = Me._boldedDays
                End If
            End If
            Return _dayTemplateSelector
        End Get
    End Property
    Private Sub Current_SizeChanged(sender As Object, e As
Windows.UI.Core.WindowSizeChangedEventArgs)
        UpdateViewState()
    End Sub
    Private Sub UpdateViewState()
        Calendar.ClearValue(HeightProperty)
        Select Case ApplicationView.Value
            Case ApplicationViewState.Filled
                Calendar.DaySlotTemplateSelector = DayTemplateSelector
                VisualStateManager.GoToState(Me, "Fill", False)
                Exit Select
            Case ApplicationViewState.FullScreenLandscape
                Calendar.DaySlotTemplateSelector = DayTemplateSelector
                VisualStateManager.GoToState(Me, "Full", False)
                Exit Select
            Case ApplicationViewState.Snapped
                ' あまりスペースがないので、デフォルトの DaySlotTemplateSelector を使用します
                Calendar.Height = 400
                Calendar.DaySlotTemplateSelector =
TryCast(Resources("SmallDaySlotTemplateSelector"), DataTemplateSelector)
                VisualStateManager.GoToState(Me, "Snapped", False)
                Exit Select
            Case ApplicationViewState.FullScreenPortrait
                Calendar.DaySlotTemplateSelector = DayTemplateSelector
                VisualStateManager.GoToState(Me, "Portrait", False)
                Exit Select
            Case Else
                Return
        End Select
        Calendar.UpdateLayout()
    End Sub
    ''' <summary>
    ''' このページが Frame に表示されるときに呼び出されます。
    ''' </summary>
    ''' <param name="e">このページにどのように到達したかを説明するイベントデータ。
    ''' Parameter プロパティは通常、ページの構成に使用されます。</param>
    Protected Overrides Sub OnNavigatedTo(e As NavigationEventArgs)
        UpdateViewState()
        _loaded = True
    End Sub
    Protected Overrides Sub OnNavigatedFrom(e As NavigationEventArgs)
        _loaded = False
        MyBase.OnNavigatedFrom(e)
End Sub
```

C# コードの書き方

C#

```
void Calendar_SelectedDateChanged(object sender, DateChangedEventArgs e)
        {
            if (Calendar.SelectedDates.Count > 0 &&
_boldedDays.ContainsKey(Calendar.SelectedDates[0]))
            {
                dayInfo.Text = _boldedDays[Calendar.SelectedDates[0]];
            }
            else
            {
                dayInfo.Text = "";
            }
        }
        /// <summary>
        /// 日付を太字にするためのカスタム DataTemplate を使用する DayTemplateSelector
        /// </summary>
        private DaySlotTemplateSelector DayTemplateSelector
        {
            get
            {
                if (_dayTemplateSelector == null)
                {
                    _dayTemplateSelector = Resources["DaySlotTemplateSelector"]
as DaySlotTemplateSelector;
                    if (_dayTemplateSelector != null)
                    {
                        _dayTemplateSelector.BoldedDays = this._boldedDays;
                    }
                }
                return _dayTemplateSelector;
            }
        }
        void Current_SizeChanged(object sender,
Windows.UI.Core.WindowSizeChangedEventArgs e)
        {
            UpdateViewState();
        }
        private void UpdateViewState()
        {
            Calendar.ClearValue(HeightProperty);
            switch (ApplicationView.Value)
            {
                case ApplicationViewState.Filled:
                    Calendar.DaySlotTemplateSelector = DayTemplateSelector;
                    VisualStateManager.GoToState(this, "Fill", false);
                    break;
                case ApplicationViewState.FullScreenLandscape:
                    Calendar.DaySlotTemplateSelector = DayTemplateSelector;
                    VisualStateManager.GoToState(this, "Full", false);
                    break;
```

```
                case ApplicationViewState.Snapped:
                    // あまりスペースがないので、デフォルトの DaySlotTemplateSelector を使
用します
                    Calendar.Height = 400;
                    Calendar.DaySlotTemplateSelector =
Resources["SmallDaySlotTemplateSelector"] as DataTemplateSelector;
                    VisualStateManager.GoToState(this, "Snapped", false);
                    break;
                case ApplicationViewState.FullScreenPortrait:
                    Calendar.DaySlotTemplateSelector = DayTemplateSelector;
                    VisualStateManager.GoToState(this, "Portrait", false);
                    break;
                default:
                    return;
            }
            Calendar.UpdateLayout();
        }
        /// <summary>
        /// このページが Frame に表示されるときに呼び出されます。
        /// </summary>
        /// <param name="e">このページにどのように到達したかを説明するイベントデータ。
        /// Parameter プロパティは通常、ページの構成に使用されます。</param>
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            UpdateViewState();
            _loaded = true;
        }
        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            _loaded = false;
            base.OnNavigatedFrom(e);
        }
```

6. **MainPage** クラスの直後に次のクラスを追加します。

Visual Basic コードの書き方

Visual Basic

```
Public Class DaySlotTemplateSelector
    Inherits C1.Xaml.Calendar.DaySlotTemplateSelector
    Public BoldedDays As New Dictionary(Of DateTime, String)()
    Protected Overrides Function SelectTemplateCore(item As Object, container As
DependencyObject) As DataTemplate
        Dim slot As DaySlot = TryCast(item, DaySlot)
        If slot IsNot Nothing AndAlso BoldedDays.ContainsKey(slot.[Date]) Then
            ' 日の DataTemplate に予定情報を表示できるようにタグに予定情報を格納します
            slot.Tag = BoldedDays(slot.[Date])
        Else
            ' 予定情報をクリアします
            slot.Tag = Nothing
        End If
        If slot IsNot Nothing AndAlso Not slot.IsAdjacent AndAlso slot.DayOfWeek
= DayOfWeek.Saturday Then
```

```
            ' 土曜日の色を設定します
            DirectCast(container, Control).Foreground = New
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255))
        End If
        ' 基本クラスでは、DaySlotTemplateSelector.Resources コレクション内に定義されたカスタム DataTemplate が選択されます(MainPage.xaml ファイルを参照)
        Return MyBase.SelectTemplateCore(item, container)
    End Function
End Class
Public Class SmallDaySlotTemplateSelector
    Inherits C1.Xaml.Calendar.DaySlotTemplateSelector
    Protected Overrides Function SelectTemplateCore(item As Object, container As
DependencyObject) As DataTemplate
        Dim slot As DaySlot = TryCast(item, DaySlot)
        If slot IsNot Nothing AndAlso Not slot.IsAdjacent AndAlso slot.DayOfWeek
= DayOfWeek.Saturday Then
            ' 土曜日の色を設定します
            DirectCast(container, Control).Foreground = New
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255))
        End If
        ' 基本クラスでは、DaySlotTemplateSelector.Resources コレクション内に定義されたカスタム DataTemplate が選択されます(MainPage.xaml ファイルを参照)
        Return MyBase.SelectTemplateCore(item, container)
    End Function
End Class
Public Class DayOfWeekTemplateSelector
    Inherits DataTemplateSelector
    Protected Overrides Function SelectTemplateCore(item As Object, container As
DependencyObject) As DataTemplate
        Dim slot As DayOfWeekSlot = TryCast(item, DayOfWeekSlot)
        If slot IsNot Nothing AndAlso slot.DayOfWeek = DayOfWeek.Saturday Then
            ' 土曜日の色を設定します
            DirectCast(container, Control).Foreground = New
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255))
        End If
        ' DataTemplate を変更しないでください
        Return Nothing
    End Function
End Class
```

C# コードの書き方

C#

```
public class DaySlotTemplateSelector : C1.Xaml.Calendar.DaySlotTemplateSelector
    {
        public Dictionary BoldedDays = new Dictionary();
        protected override DataTemplate SelectTemplateCore(object item,
DependencyObject container)
        {
            DaySlot slot = item as DaySlot;
            if (slot != null && BoldedDays.ContainsKey(slot.Date))
            {
```

```
                // 日の DataTemplate に予定情報を表示できるようにタグに予定情報を格納します
                slot.Tag = BoldedDays[slot.Date];
            }
            else
            {
                // 予定情報をクリアします
                slot.Tag = null;
            }
            if (slot != null && !slot.IsAdjacent && slot.DayOfWeek ==
DayOfWeek.Saturday)
            {
                // 土曜日の色を設定します
                ((Control)container).Foreground = new
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255));
            }
            // 基本クラスでは、DaySlotTemplateSelector.Resources コレクション内に定義された
カスタム DataTemplate が選択されます(MainPage.xaml ファイルを参照)
            return base.SelectTemplateCore(item, container);
        }
    }
    public class SmallDaySlotTemplateSelector :
C1.Xaml.Calendar.DaySlotTemplateSelector
    {
        protected override DataTemplate SelectTemplateCore(object item,
DependencyObject container)
        {
            DaySlot slot = item as DaySlot;
            if (slot != null && !slot.IsAdjacent && slot.DayOfWeek ==
DayOfWeek.Saturday)
            {
                // 土曜日の色を設定します
                ((Control)container).Foreground = new
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255));
            }
            // 基本クラスでは、DaySlotTemplateSelector.Resources コレクション内に定義された
カスタム DataTemplate が選択されます(MainPage.xaml ファイルを参照)
            return base.SelectTemplateCore(item, container);
        }
    }
    public class DayOfWeekTemplateSelector : DataTemplateSelector
    {
        protected override DataTemplate SelectTemplateCore(object item,
DependencyObject container)
        {
            DayOfWeekSlot slot = item as DayOfWeekSlot;
            if (slot != null && slot.DayOfWeek == DayOfWeek.Saturday)
            {
                // 土曜日の色を設定します
                ((Control)container).Foreground = new
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255));
            }
            // DataTemplate を変更しないでください
            return null;
```

```
        }
    }
```

次の「手順3：Calendar アプリケーションの実行」では、**Calendar for UWP** の機能について説明します。

ここまでの成果

これで、**C1Calendar** にデータを追加できました。次の手順では、実行時の動作をいくつか確認します。

手順3：Calendar アプリケーションの実行

これまでに、UWP スタイルアプリケーションを作成し、外観と動作をカスタマイズしたので、次にアプリケーションを実行します。アプリケーションを実行し、**Calendar for UWP** の実行時の動作を確認するには、次の手順に従います。

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

   アプリケーションで定義した書式設定とイベントがどのように表示されるかに注目してください。
2. カレンダーの上部にある**前へ**または**次へ**矢印ボタンをクリックして、前の月または次の月に移動します。
3. 複数の日を選択するには、**[Ctrl]**キーまたは**[Shift]**キーを押しながら項目をクリックします。
4. また、月ヘッダーをタップして**年カレンダーモード**に変更できます。そして、年ヘッダーをタップすると**10年カレンダー**に変更できます。
5. 10年カレンダーモードを使用しているときには、ある年をクリックして年カレンダーモードに戻られます。また、年カレンダーモードを使用しているときには、ある月をクリックして月カレンダーに戻られます。

ここまでの成果

おめでとうございます。これで **Calendar for UWP** クイックスタートは完了です。Calendar コントロールを使用するアプリケーションを作成し、アプリケーションの実行時機能をいくつか確認することができました。

タスク別ヘルプ

タスク別ヘルプセクションは、Visual Studio.NET 環境でのプログラミングにある程度慣れていることを前提としています。

C1Calendar への太字の日付の追加

BoldedDates プロパティを使用して、**C1Calendar** コントロールに太字の日付を追加します。次の手順に従います。

Visual Basic コードの書き方

Visual Basic

```
' 太字の日を追加します
cal1.BoldedDates.Add(DateTime.Today.AddDays(2))
cal1.BoldedDates.Add(DateTime.Today.AddDays(12))
cal1.BoldedDates.Add(DateTime.Today.AddDays(22))
cal1.BoldedDates.Add(DateTime.Today.AddDays(-2))
cal1.BoldedDates.Add(DateTime.Today.AddDays(-12))
cal1.BoldedDates.Add(DateTime.Today.AddDays(-22))
```

C# コードの書き方

C#

```
// 太字の日を追加します
cal1.BoldedDates.Add(DateTime.Today.AddDays(2));
cal1.BoldedDates.Add(DateTime.Today.AddDays(12));
cal1.BoldedDates.Add(DateTime.Today.AddDays(22));
cal1.BoldedDates.Add(DateTime.Today.AddDays(-2));
cal1.BoldedDates.Add(DateTime.Today.AddDays(-12));
cal1.BoldedDates.Add(DateTime.Today.AddDays(-22));
```

DaySlotTemplateSelector を使用して日をカスタマイズする

日曜日と今日の日付の色をカスタマイズするには、次のコードを追加します。

Visual Basic コードの書き方

Visual Basic

```
' 日曜日を赤色で表示し、今日の日付を緑色で表示します
Dim datesSelector As DaySlotTemplateSelector =
TryCast(Me.Resources("DaySlotTemplateSelector"), DaySlotTemplateSelector)
cal1.DaySlotTemplateSelector = datesSelector
' DaySlotTemplateSelector クラスインスタンス内で定義した太字の日の辞書を使用します
Me._boldedDays = datesSelector.BoldedDays
cal1.DayOfWeekSlotTemplateSelector = New DayOfWeekTemplateSelector()
cal1.WeekendBrush = New SolidColorBrush(Colors.Red)
cal1.TodayBrush = New SolidColorBrush(Colors.Green)
```

C# コードの書き方

C#

```
// 日曜日を赤色で表示し、今日の日付を緑色で表示します
DaySlotTemplateSelector datesSelector = this.Resources["DaySlotTemplateSelector"] as
DaySlotTemplateSelector;
cal1.DaySlotTemplateSelector = datesSelector;
// DaySlotTemplateSelector クラスインスタンス内で定義した太字の日の辞書を使用します
this._boldedDays = datesSelector.BoldedDays;
cal1.DayOfWeekSlotTemplateSelector = new DayOfWeekTemplateSelector();
cal1.WeekendBrush = new SolidColorBrush(Colors.Red);
cal1.TodayBrush = new SolidColorBrush(Colors.Green);}
```