ComponentOne

Calendar for UWP

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: sales@componentone.com
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

Table of Contents

Calendar for UWP

Create date-driven dashboards and scheduling apps with **Calendar for UWP**. The C1Calendar control can be used for date navigation, date range selection, and it can display custom calendar information such as appointments. It can be used to display a single month and select any number of days. Highlight days with simple formatting or custom content.

The following topics will get you started with **Calendar for UWP**.

Getting Started with UWP Edition

Help with UWP Edition

Getting Started

For information on installing **ComponentOne Studio UWP Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with ComponentOne Studio UWP Edition.

Key Features

Calendar for UWP Edition allows you to create customized, rich applications. Make the most of it by taking advantage of the following key features:

- **Gesture-based Month Navigation**

  By default, C1Calendar supports slide and flick gestures for month-to-month navigation. Or tap the navigation buttons to achieve the same effect.

- **Date Range Selection**

  Enable the user to select one day or many days through a default tap or hold and slide gesture. Control the number of days they can select using the MaxSelectionCount property.

- **Customizable Calendar Settings**

  Specify the starting day of the week, the work weekdays collection, and more. You can use **C1Calendar** with any supported culture in UWP. Visually distinguish between work days and weekends through special brush properties. Make any date bolded to highlight it to the end-user.

- **Easy and Flexible Styling Model**

  With **C1Calendar** you can easily change control brushes without having to override templates. Each visible part of the control has its own brush, including adjacent days, weekends, selected days, the month header and more.

- **Customize Date Appearance and Content**

  You can alter the appearance of any individual day using a custom template and template selector. Use custom **C1DataTemplateSelector** implementations to display custom content within the date blocks, such as calendar appointments.

- **Show Week Numbers**

  Display week numbers by just setting one simple property. There are never any trailing empty weeks because **C1Calendar** always displays the minimum number of rows required per month whether it is 4, 5 or 6 weeks long.

- **Year and Decade Modes**

  Enter historical dates (dates from previous years or decades) easily using the year mode and decade mode. Tap the month header to switch to the year mode and the year header to switch to the decade mode. See Step 3 of 3: Running the Calendar Application for further details on how to use this feature.

Calendar for UWP Quick Start

The following quick start guide is intended to get you up and running with **Calendar for UWP**. In this quick start you'll start in Visual Studio and create a new project, add **Calendar for UWP** controls to your application, and customize the appearance and behavior of the controls.

Step 1 of 3: Creating an Application with C1Calendar Control

In this step, you'll create a UWP application in Visual Studio using **Calendar for UWP**.

Complete the following steps:

1. In Visual Studio 2012 Select **File | New | Project**.
2. In the **New Project** dialog box, expand a language in the left pane, under the language select **Windows Store**, and in the templates list select **Blank App (XAML)**. Enter a **Name** and click **OK** to create your project.
3. In MainPage.xaml, add the following <Page.Resources> markup between the <Page>and <Grid> tags to customize the control:

   To write the markup in XAML:

Markup

```
<Page.Resources>
    <!-- return custom DataTemplates for calendar days -->
    <local:DaySlotTemplateSelector x:Key="DaySlotTemplateSelector">
        <Calendar:DaySlotTemplateSelector.Resources>
            <ResourceDictionary>
                <DataTemplate x:Key="BoldedDay">
                    <Grid>
                        <Grid.RowDefinitions>
                            <RowDefinition />
                            <RowDefinition Height="Auto"/>
                        </Grid.RowDefinitions>
                        <!-- show appointments information saved in the
DaySlot.Tag property -->
                        <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. Navigate to the Toolbox and double-click the C1Calendar icon to add the control to the grid. This will add the reference and XAML namespace automatically. The XAML markup resembles the following:

Markup

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

5. Add the markup in <Calendar:C1Calendar> tag to customize the control:

Markup

```
<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>
```

This gives the control a name and customizes the formatting and appearance of the calendar. Note that you'll add code for the referenced event handler in a later step.

6. Add the following markup above the Calendar and between the <Grid Background="{StaticResource ApplicationPageBackgroundThemeBrush}"> and <Calendar:C1Calendar> tags:

To write the markup in XAML:

Markup

```
<!--App Orientation States-->
        <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. Add the following markup below the Calendar and between the <Calendar:C1Calendar> and </Grid> tags:

   To write the markup in XAML:

Markup

```
<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>
```

What You've Accomplished

You've successfully created a UWP style application containing a C1Calendar control. In the next step, Step 2 of 3: Adding Data to the Calendar, you will add the data for **C1Calendar**.

Step 2 of 3: Adding Data to the Calendar

In the last step, you added the C1Calendar control to the application. In this step, you will add a **DataSeries** object and data for it.

Complete the following steps to add data to the calendar programmatically:

1. Select **View | Code** to switch to Code view.
2. Add the following imports statements to the top of the page:

   To write the code in Visual Basic:

   Visual Basic

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

   To write the code in C#:

   C#

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

3. Add the following code just inside the **MainPage** class:

   To write the code in Visual Basic:

   Visual Basic

   ```
   ' dictionary of appointments
   Private _boldedDays As New Dictionary(Of DateTime, String)()
   Private _dayTemplateSelector As DaySlotTemplateSelector = Nothing
   Private _loaded As Boolean = False
   ```

   To write the code in C#:

   C#

   ```
   // dictionary of appointments
   private Dictionary<DateTime, string> _boldedDays = new Dictionary<DateTime,
   string>();
   private DaySlotTemplateSelector _dayTemplateSelector = null;
   private bool _loaded = false;
   ```

4. Update the **MainPage** code so it appears like the following:

To write the code in Visual Basic:

```
Visual Basic
Public Sub New()
    Me.InitializeComponent()
    AddHandler Window.Current.SizeChanged, AddressOf Current_SizeChanged
    Calendar.DayOfWeekSlotTemplateSelector = New DayOfWeekTemplateSelector()

    ' add some bold days
    _boldedDays.Add(DateTime.Today.AddDays(2), "Game" & vbCr & vbLf & "Don't
forget!")
    _boldedDays.Add(DateTime.Today.AddDays(13), "Birthday")
    _boldedDays.Add(DateTime.Today.AddDays(22), "Important Meeting")
    _boldedDays.Add(DateTime.Today.AddDays(-1), "Anniversary" & vbCr & vbLf &
"Party at 8")
    _boldedDays.Add(DateTime.Today.AddDays(-12), "Doctor's Appointment")
    _boldedDays.Add(DateTime.Today.AddDays(-21), "Conference Day 2")
    _boldedDays.Add(DateTime.Today.AddDays(-22), "Conference Day 1")
    For Each val As DateTime In _boldedDays.Keys
        Calendar.BoldedDates.Add(val)
    Next
End Sub
```

To write the code in C#:

```
C#
public MainPage()
{
    this.InitializeComponent();
    Window.Current.SizeChanged += Current_SizeChanged;
    Calendar.DayOfWeekSlotTemplateSelector = new DayOfWeekTemplateSelector();

    // add some bold days
    _boldedDays.Add(DateTime.Today.AddDays(2), "Game\r\nDon't forget!");
    _boldedDays.Add(DateTime.Today.AddDays(13), "Birthday");
    _boldedDays.Add(DateTime.Today.AddDays(22), "Important Meeting");
    _boldedDays.Add(DateTime.Today.AddDays(-1), "Anniversary\r\nParty at 8 ");
    _boldedDays.Add(DateTime.Today.AddDays(-12), "Doctor's Appointment");
    _boldedDays.Add(DateTime.Today.AddDays(-21), "Conference Day 2");
    _boldedDays.Add(DateTime.Today.AddDays(-22), "Conference Day 1");
    foreach (DateTime val in _boldedDays.Keys)
    {
        Calendar.BoldedDates.Add(val);
    }
}
```

5. Add the following code just under the code you just added, still in the **MainPage** class:

To write the code in Visual Basic:

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>
    ''' The DayTemplateSelector to use custom DataTemplate for bolded dates
    ''' </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
                ' we have too few space, so use default DaySlotTemplateSelector
                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>
    ''' Invoked when this page is about to be displayed in a Frame.
    ''' </summary>
    ''' <param name="e">Event data that describes how this page was reached.
The Parameter
    ''' property is typically used to configure the page.</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
```

To write the code in C#:

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>
        /// The DayTemplateSelector to use custom DataTemplate for bolded dates
        /// </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:
                    // we have too few space, so use default
DaySlotTemplateSelector
                    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>
        /// Invoked when this page is about to be displayed in a Frame.
        /// </summary>
        /// <param name="e">Event data that describes how this page was reached.
The Parameter
        /// property is typically used to configure the page.</param>
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            UpdateViewState();
            _loaded = true;
        }

        protected override void OnNavigatedFrom(NavigationEventArgs e)
        {
            _loaded = false;
            base.OnNavigatedFrom(e);
        }
```

6. Add the following classes just under the **MainPage** class:

To write the code in Visual Basic:

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
            ' put appointments information into tag, so that it is possible to
show it in a day DataTemplate
            slot.Tag = BoldedDays(slot.[Date])
        Else
            ' clear appointments information
            slot.Tag = Nothing
        End If
        If slot IsNot Nothing AndAlso Not slot.IsAdjacent AndAlso slot.DayOfWeek
= DayOfWeek.Saturday Then
            ' set color for Saturday
            DirectCast(container, Control).Foreground = New
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255))
        End If
        ' the base class will select custom DataTemplate, defined in the
```

```
DaySlotTemplateSelector.Resources collection (see MainPage.xaml file)
        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
            ' set color for Saturday
            DirectCast(container, Control).Foreground = New
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255))
        End If
        ' the base class will select custom DataTemplate, defined in the
DaySlotTemplateSelector.Resources collection (see MainPage.xaml file)
        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
            ' set color for Saturday
            DirectCast(container, Control).Foreground = New
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255))
        End If
        ' don't change DataTemplate at all
        Return Nothing
    End Function

End Class
```

To write the code in C#:

C#

```
public class DaySlotTemplateSelector : C1.Xaml.Calendar.DaySlotTemplateSelector
    {
        public Dictionary<DateTime, string> BoldedDays = new
Dictionary<DateTime, string>();

        protected override DataTemplate SelectTemplateCore(object item,
DependencyObject container)
        {
            DaySlot slot = item as DaySlot;
            if (slot != null && BoldedDays.ContainsKey(slot.Date))
```

```
            {
                // put appointments information into tag, so that it is possible
to show it in a day DataTemplate
                slot.Tag = BoldedDays[slot.Date];
            }
            else
            {
                // clear appointments information
                slot.Tag = null;
            }
            if (slot != null && !slot.IsAdjacent && slot.DayOfWeek ==
DayOfWeek.Saturday)
            {
                // set color for Saturday
                ((Control)container).Foreground = new
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255));
            }
            // the base class will select custom DataTemplate, defined in the
DaySlotTemplateSelector.Resources collection (see MainPage.xaml file)
            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)
            {
                // set color for Saturday
                ((Control)container).Foreground = new
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255));
            }
            // the base class will select custom DataTemplate, defined in the
DaySlotTemplateSelector.Resources collection (see MainPage.xaml file)
            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)
            {
```

```
                // set color for Saturday
                ((Control)container).Foreground = new
SolidColorBrush(Windows.UI.Color.FromArgb(255, 0, 90, 255));
            }
            // don't change DataTemplate at all
            return null;
        }

    }
```

In the next step, you'll examine the features of **Calendar for UWP**.

What You've Accomplished

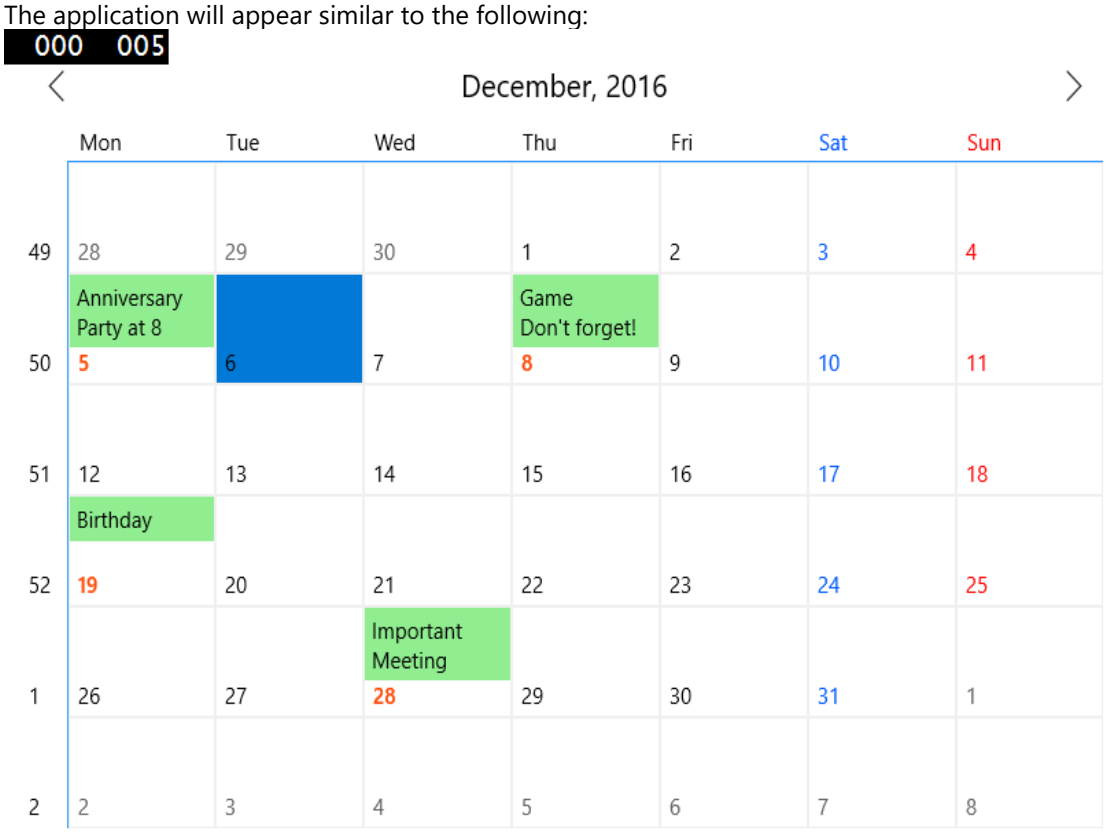
You have successfully added data to **C1Calendar**. In the next step you'll observe some run-time behaviors.

Step 3 of 3: Running the Calendar Application

Now that you've created a UWP style application and customized the application's appearance and behavior, the only thing left to do is run your application. To run your application and observe **Calendar for UWP Edition's** run-time behavior, complete the following steps:

1. From the **Debug** menu, select **Start Debugging** to view how your application will appear at run time.

   The application will appear similar to the following:



   Notice how the formatting and events defined in your application appear.

2. Navigate to a previous or next month by clicking the **Previous** or **Next** arrow buttons at the top of the calendar.
3. To select multiple days, hold the CTRL or SHIFT key and click items.

4. Tap the month header to switch to the year mode. Tap the year header to switch to the decade mode.
5. In the decade mode, select a year to switch back to the year mode and in the year mode select a month to switch back to the month mode.

What You've Accomplished

Congratulations! You've completed the **Calendar for UWP** quick start and created an application using the Calendar control and viewed some of the run-time capabilities of your application.

Calendar for UWP Task-Based Help

The task-based help section assumes that you are familiar with programming in the Visual Studio environment.

Adding Bolded Dates to the C1Calendar

Use the C1Calendar.BoldedDates property to add bolded dates to the C1Calendar control like the following:

To write the code in Visual Basic:

Visual Basic

```
' add some bold days
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))
```

To write the code in C#:

C#

```
// add some bold days
   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));
```

Customizing Days Using DaySlotTemplateSelector

To customize the color of Sundays' and today's date, add the following:

To write the code in Visual Basic:

Visual Basic

```
' show Sundays in Red and Today’s date in Green
Dim datesSelector As DaySlotTemplateSelector =
TryCast(Me.Resources("DaySlotTemplateSelector"), DaySlotTemplateSelector)
cal1.DaySlotTemplateSelector = datesSelector
' use bolded days dictionary defined in the DaySlotTemplateSelector class instance
Me._boldedDays = datesSelector.BoldedDays
cal1.DayOfWeekSlotTemplateSelector = New DayOfWeekTemplateSelector()
cal1.WeekendBrush = New SolidColorBrush(Colors.Red)
cal1.TodayBrush = New SolidColorBrush(Colors.Green)
```

To write the code in C#:

C#

```
// show Sundays in Red and Today's Date in Green
            DaySlotTemplateSelector datesSelector =
this.Resources["DaySlotTemplateSelector"] as DaySlotTemplateSelector;
            cal1.DaySlotTemplateSelector = datesSelector;
            // use bolded days dictionary defined in the DaySlotTemplateSelector
class instance
            this._boldedDays = datesSelector.BoldedDays;
            cal1.DayOfWeekSlotTemplateSelector = new DayOfWeekTemplateSelector();
            cal1.WeekendBrush = new SolidColorBrush(Colors.Red);
            cal1.TodayBrush = new SolidColorBrush(Colors.Green);
}
```