Scheduler for UWP

2018.04.10 更新

グレープシティ株式会社

Scheduler for UWP

目次

Scheduler for UWP

Scheduler for UWP を使用して、独自のスケジュールアプリケーションを作成できます。Microsoft Outlook や Windows 8 Calendar アプリケーションとまったく同様に、アラーム、ラベル、空き状況などを使用して、1回、全日、または定期的な予定をスケジュールできます。簡単なデータ連結、インポート/エクスポート、タッチサポートを使用して、完全なスケジュールアプリケーションの開発がこれまで以上に容易になりました。

C1Scheduler コントロールは、Windows 8 Calendar に似た予定の表示/編集機能を提供します。

主な特長

Scheduler for UWP には、次の機能があります。

- **4つの組み込みビュー**

  C1Scheduler コントロールには、4つの組み込みビューがあります。これらのビューを使用して、スケジュールをさまざまな方法で表示する機能をユーザーに提供できます。組み込まれているビュータイプは、日、週、週間勤務日、および月です。このコントロールは、カスタムビューの作成もサポートします。

- **ジェスチャベースのナビゲーション**

  C1Scheduler コントロールは、タッチ中心の没入型 Windows アプリケーション向けに作成されており、ジェスチャベースの日付ナビゲーションを提供します。ユーザーは左右にスワイプするだけで、現在の月、週、または日を変更できます。ナビゲーションボタンやスクロールバーが付いているため、デスクトップユーザーやマウスユーザーにも最適です。予定をドラッグして期間や開始時刻を変更することもできます。

- **完全な予定エディタ**

  C1Scheduler は、多くのコードを記述することなく使用できる完全な予定エディタを提供します。予定エディタをさらに詳細にカスタマイズして、アプリケーション固有の要件を満たすことができます。

- **データ連結**

  コントロールを任意の IEnumerable データソースに連結できます。さらに、アプリケーションを次回実行するまで予定データを XML として保持できます。

- **データのインポート/エクスポート**

  XML 形式と iCalendar（iCal）形式でデータを保存またはロードできます。

- **ビューのカスタマイズ**

  組み込みビュー以外にも、カスタムスケジュールビューを作成して使用することができます。たとえば、日を縦方向に積み重ねた週ビューを作成できます。

- **アラーム**

  C1Scheduler は、予定が発生する前の指定された時間にアラームダイアログボックスを表示できます。ユーザーは、いくつかのアラームを終了するか、予定を開くか、アラームをリセットして後で再表示（スヌーズ）することができます。

- **ライト、ダーク、ハイコントラストのテーマ**

  Windows の組み込みテーマを使用すると、アプリケーションのスタイルに合わせる作業が必要ありません。もちろん、Visual Studio でブラシプロパティを設定するだけで、カスタムテーマを作成することもできます。

クイックスタート

手順1:アプリケーションの設定

この手順では、Windows ストアアプリケーションを設定し、C1Scheduler コントロールを追加します。UWP スタイルアプリケーションの作成の詳細については、「UWP スタイルアプリケーションの作成」を参照してください。

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。これにより、**[新しいプロジェクト]**ダイアログウィンドウが開きます。
2. 左ペインで、**[テンプレート]**セクションからプログラミング言語を選択し、ドロップダウン矢印をクリックします。**[ストアアプリ]**を選択してから、**[Windows アプリ]**を選択します。
3. インストール済みテンプレートリストから**[空白のアプリ(ユニバーサルWindows)]**を選択します。
4. アプリケーションに名前を付け、**[OK]**をクリックします。アプリケーションが開きます。
5. ソリューションエクスプローラーで、アプリケーションを見つけます。**References** フォルダを右クリックし、コンテキストメニューから**[新しい参照の追加]**を選択します。
6. **[参照マネージャー]**で、**[Windows 8.1]**セクションのドロップダウン矢印を選択し、**[拡張]**を選択します。中央ペインに **ComponentOne for UWP アセンブリ**が表示されます。次のアセンブリを選択し、**[OK]**をクリックします。

   - C1.UWP
   - C1.UWP.Calendar
   - C1.UWP.DateTimeEditors
   - C1.UWP.Schedule

7. **MainPage.xaml** ページをダブルクリックして開きます。次の XAML 名前空間を開始タグ <Page> に追加します。

   マークアップ
   ```
   xmlns:Schedule="using:C1.Xaml.Schedule"
   ```

   <Page> タグ全体は次のようになります。

   マークアップ
   ```
   <Page
     x:Class="SchedTest2.MainPage"
     xmlns:Schedule="using:C1.Xaml.Schedule"
     xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
     xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
     xmlns:local="using:SchedTest2"
     xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
     xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
     mc:Ignorable="d">
   ```

8. <Grid> タグに次のマークアップを追加して、アプリケーションに C1Scheduler コントロールを追加します。

   マークアップ
   ```
   <Schedule:C1Scheduler Name="sched1" ViewType="Month" Grid.Row="1" Height="500" Width="650">
   </Schedule:C1Scheduler>
   ```

ここまでの成果

このトピックでは、Windows ストアアプリケーションを作成し、そのアプリケーションにアセンブリ参照と C1Scheduler コントロールを追加しました。次の手順では、アプリケーションにさらに XAML マークアップを追加し、コントロールのビューを制御す

るためのボタンを追加します。

手順2:ボタンの追加

この手順では、アプリケーションにさらに XAML マークアップを追加します。このマークアップは、C1Scheduler のビューを制御するためのボタン要素を作成します。

1. 開始タグ <Grid> の下に次の XAML マークアップを追加します。これは、行定義を追加します。

マークアップ

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition />
</Grid.RowDefinitions>
```

2. Grid.RowDefinitions の下に、汎用 StackPanel コントロールを追加します。このコントロールは次のマークアップのようになります。

マークアップ

```
<StackPanel Orientation="Horizontal" HorizontalAlignment="Left" VerticalAlignment="Center">
    <StackPanel.Resources>
        <Style TargetType="Button" x:Key="btnStyle">
            <Setter Property="VerticalAlignment" Value="Stretch"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="MinWidth" Value="100"/>
        </Style>
    </StackPanel.Resources>
</StackPanel>
```

3. 4つの汎用ボタンコントロールを</StackPanel.Resources> と</StackPanel> タグの間に追加します。次のようにマークアップを編集します。各ボタンには Click イベントがあります。

マークアップ

```
<Button Click="DayClick" Style="{StaticResource btnStyle}">
    <Button.Content>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="&#xe161;" FontFamily="Segoe UI Symbol" FontSize="22"
VerticalAlignment="Center"/>
            <TextBlock Text=" Day" VerticalAlignment="Center"/>
        </StackPanel>
    </Button.Content>
</Button>
<Button Click="WorkWeekClick" Style="{StaticResource btnStyle}">
    <Button.Content>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="&#xe162;" FontFamily="Segoe UI Symbol" FontSize="22"
VerticalAlignment="Center"/>
            <TextBlock Text=" Work Week" VerticalAlignment="Center"/>
        </StackPanel>
    </Button.Content>
</Button>
<Button Click="WeekClick" Style="{StaticResource btnStyle}">
```

```
    <Button.Content>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="&#xe162;" FontFamily="Segoe UI Symbol" FontSize="22"
VerticalAlignment="Center"/>
            <TextBlock Text=" Week" VerticalAlignment="Center"/>
        </StackPanel>
    </Button.Content>
</Button>
<Button Click="MonthClick" Style="{StaticResource btnStyle}">
    <Button.Content>
        <StackPanel Orientation="Horizontal">
            <TextBlock Text="&#xe163;" FontFamily="Segoe UI Symbol" FontSize="22"
VerticalAlignment="Center"/>
            <TextBlock Text=" Month" VerticalAlignment="Center"/>
        </StackPanel>
    </Button.Content>
</Button>
```

ここまでの成果

この手順では、アプリケーションに4つのボタンコントロールのマークアップを追加しました。次の手順では、**Button_Click** イベントのコードを追加します。

手順3:コードの追加

この手順では、**Button_Click** イベントのコードを追加します。

1. 次の名前空間をページの先頭に追加します。

C#

```
using C1.Xaml.Schedule;
```

2. 次のように **MainPage()** コンストラクタを編集します。

C#

```
public MainPage()
{
    this.Unloaded += MainPage_Unloaded;
    this.InitializeComponent();

    sched1.Settings.FirstVisibleTime = System.TimeSpan.FromHours(8);
}
```

3. **MainPage()** コンストラクタの後に次のコードを追加します。このコードは、各ボタンの **Click** イベントを追加します。

C#

```
void MainPage_Unloaded(object sender, RoutedEventArgs e)
{
    // メモリリークを避けるために C1Scheduler コントロールを破棄します
    sched1.Dispose();
}
private void DayClick(object sender, RoutedEventArgs e)
```

```
{
    sched1.ChangeStyle(sched1.OneDayStyle);
}
private void WorkWeekClick(object sender, RoutedEventArgs e)
{
    sched1.ChangeStyle(sched1.WorkingWeekStyle);
}
private void WeekClick(object sender, RoutedEventArgs e)
{
    sched1.ChangeStyle(sched1.WeekStyle);
}
private void MonthClick(object sender, RoutedEventArgs e)
{
    sched1.ChangeStyle(sched1.MonthStyle);
}
```

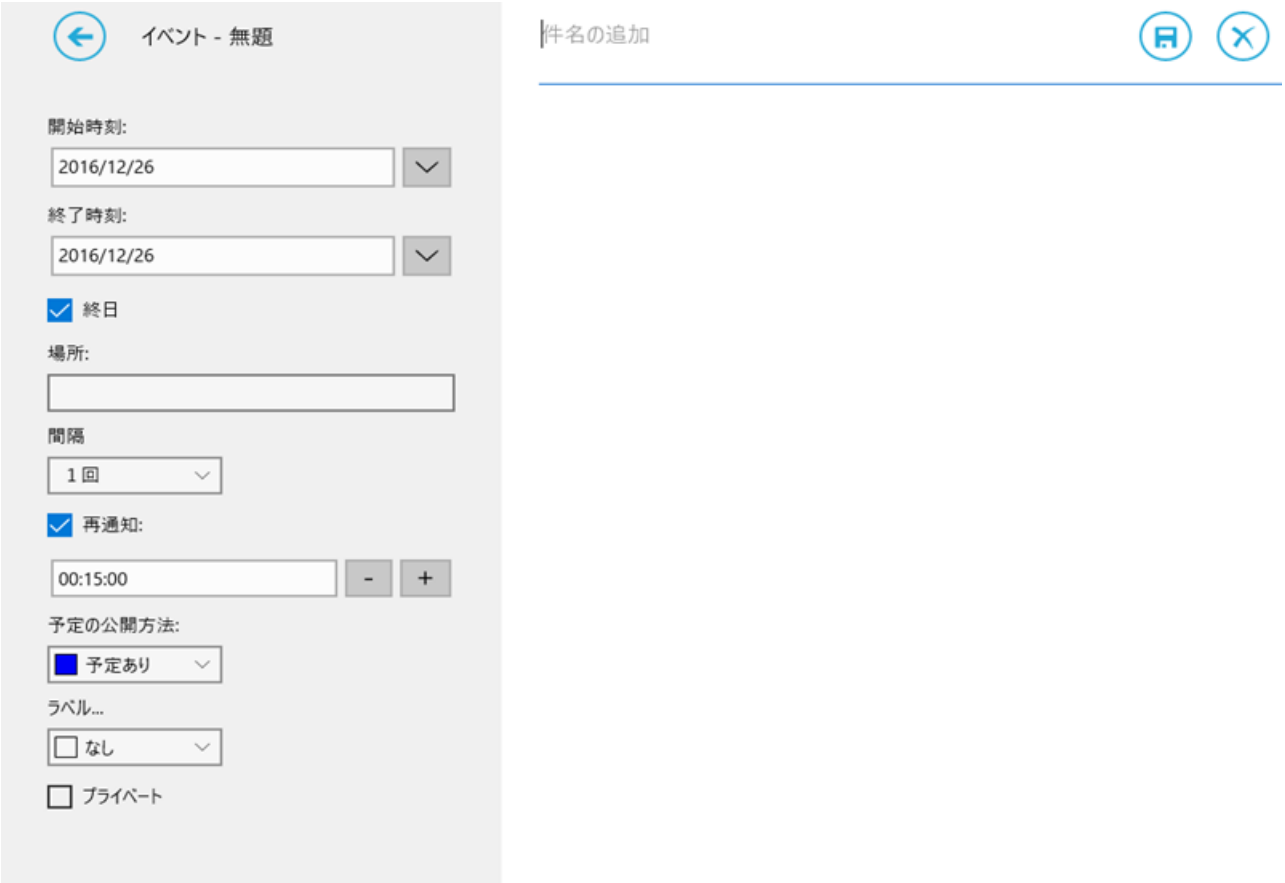
この手順では、**Button_Click** イベントを制御するコードを追加しました。次の手順では、このアプリケーションを実行します。

手順4:アプリケーションの実行

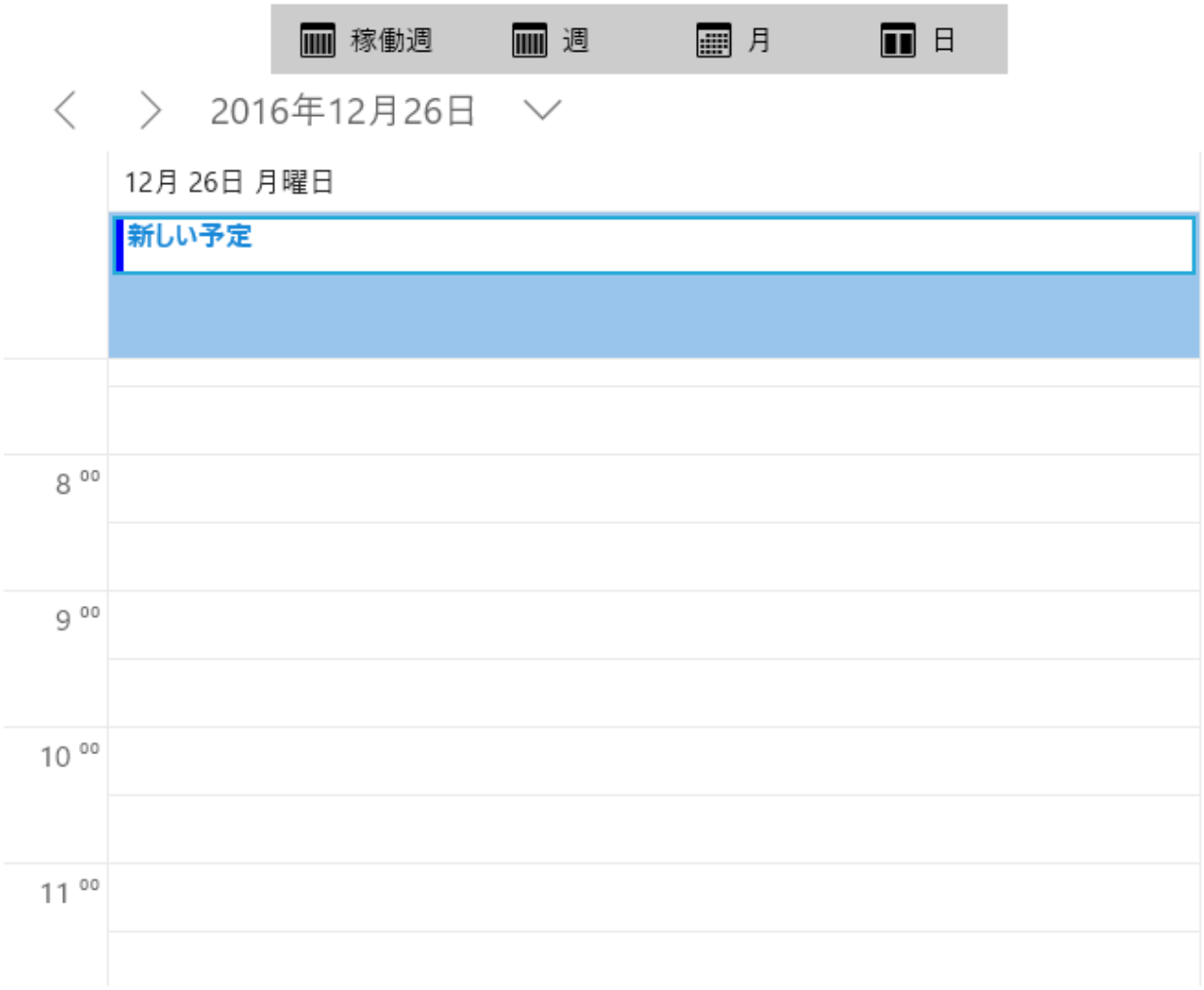
この手順ではアプリケーションを実行します。

- **[F5]**キーを押すか、デバッグを開始して、アプリケーションを実行します。次のようになります。

- 新しい予定を追加するには、カレンダーからいずれかの日をダブルタップまたはダブルクリックします。



- **[保存]**ボタンをタップまたはクリックすると、予定が C1Scheduler コントロールに追加されます。この予定は、日ビューに表示されます。



ここまでの成果

おめでとうございます。これで、**C1Scheduler クイックスタート**は終了です。このトピックでは、Windows ストアアプリケーションを作成し、C1Scheduler コントロールといくつかのボタンのマークアップを追加し、**Button_Click** イベントのコードを追加しました。

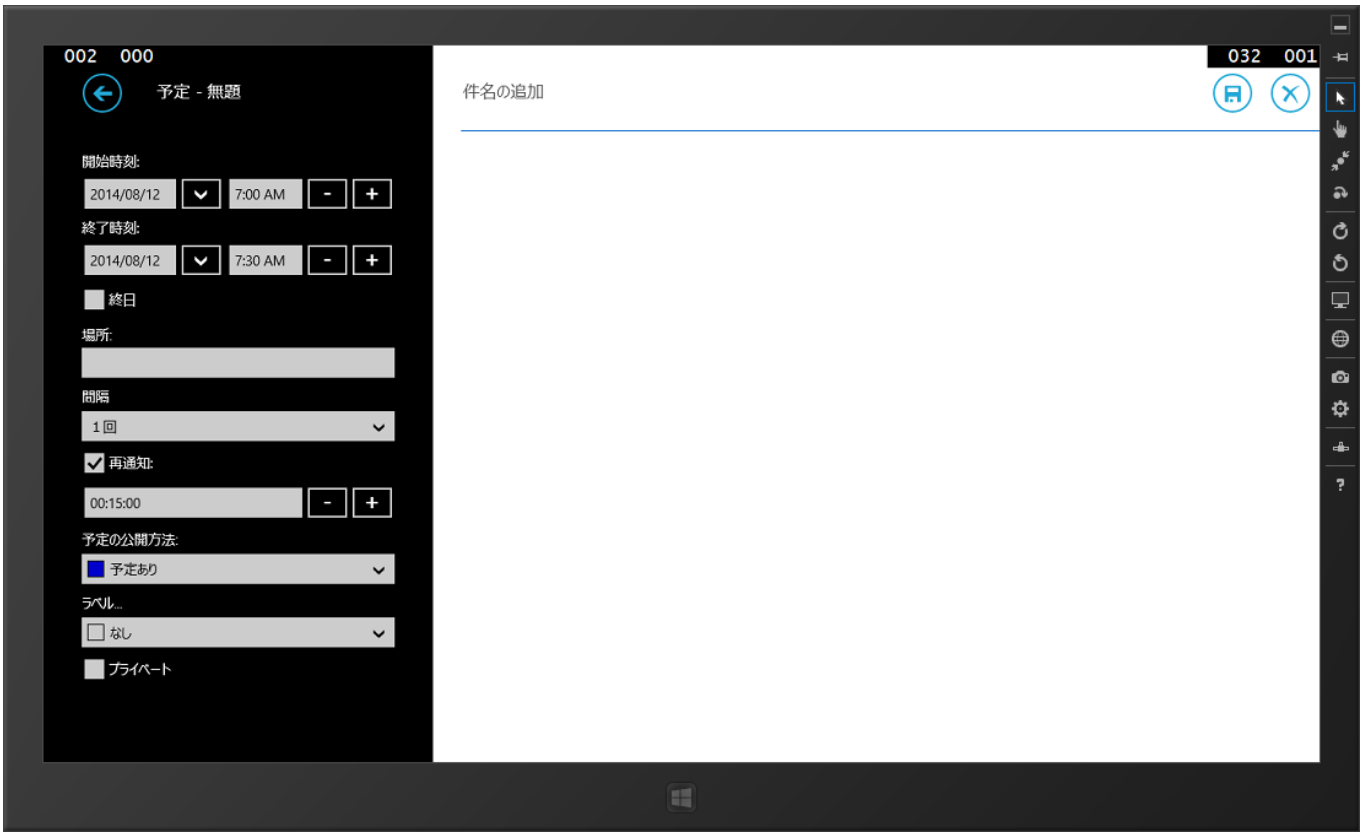
C1Scheduler の使い方

予定

1つの予定は数分から数日まで一定の期間に及び、繰り返し、場所、メモなど、その予定に関する情報を持つことができます。C1Scheduler の**日ビュー**では、予定は次のように表示されます。

新しい予定を追加するには、日または時刻をダブルクリックまたはダブルタップします。既存の予定を編集するには、それをダブルクリックまたはダブルタップします。タップまたはクリックすると、**予定ダイアログ**が開きます。

新しい予定を追加するには、日または時刻をダブルクリックまたはダブルタップします。既存の予定を編集するには、それをダブルクリックまたはダブルタップします。タップまたはクリックすると、予定ダイアログが開きます。また、日または時刻に Enter キーを押下してタイトルを指定することでも新規予定を作成できます。その予定に情報を追加するには、Enter キーを押下して予定ダイアログを開き、そのダイアログに場所、開始時刻、終了時刻などを指定します。

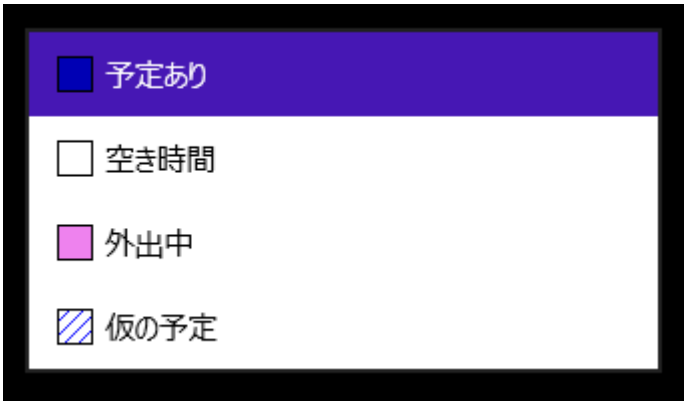


マーカーとラベル

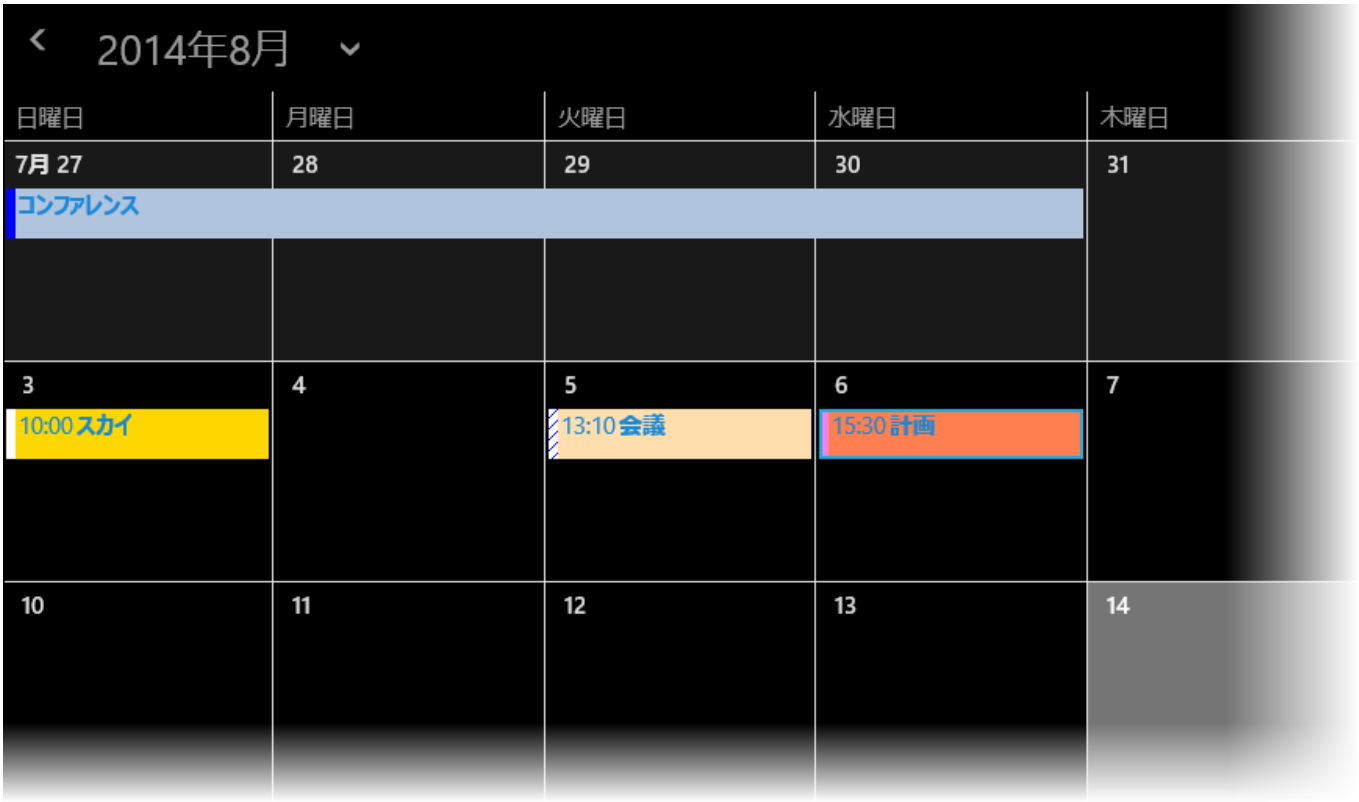
[予定の公開方法]オプションと**[ラベル]**オプションを使用して、カレンダーを詳細にカスタマイズできます。12 個のオプションを使用して、これから何が起こるかを一目でわかるように予定にラベルを付けることができます。4つのステータスマーカーを使用して状況をマークすることで、他の人々に自分の都合を確認してもらうことができます。

ラベル

ステータスマーカー



ラベルは予定の背景色を決定し、**ステータス**マーカーは予定名の左側に表示されます。次の図に、4つのラベル色とすべてのステータス色を示します。



アラーム

予定を作成したら、アラーム時刻を設定できます。その時間に、C1Scheduler コントロールはアラームポップアップを表示します。

予定の編集

Scheduler for UWP では、予定をデザイナまたは[予定の編集]ダイアログで編集や更新できます。

1. 予定の件名をデザイナ上で編集や更新するには、[F2]キーを使用します。編集したテキストを保存するには[Enter]キーと、編集をキャンセルするには [Esc] キーを使用できます。
2. [予定の編集]ダイアログを開くには、既存の予定をダブルクリックします。または、予定が選択状態で[Enter]キーを押すと、[予定の編集]ダイアログが開きます。
3. [予定の編集]ダイアログ内の対象フィールドを編集後、予定を更新するには右端の[保存]ボタンを押すか、[Ctrl+S]キーを使用ます。

また、選択状態にある予定に[Tab (Shift+Tab)]キーを使用して現在のビュー内タブの順番に従って、フォーカスや選択状態を予定間に移動できます。

[予定の編集]ダイアログのカスタマイズ

EditAppointmentControl.xaml ファイルをカスタマイズすることで、**[予定の編集]ダイアログ**を簡単にカスタマイズできます。

参考のため、EditAppointmentControl の作成に使用するマークアップとコードを次に示します。

▶ XAML でマークアップを書く場合

```
XAML
```

```
<UserControl
  x:Class="SchedulerTest3.EditAppointmentControl"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:local="using:SchedulerTest3"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  xmlns:c1="using:C1.Xaml"
  xmlns:c1datetime="using:C1.Xaml.DateTimeEditors"
  xmlns:Schedule="using:C1.Xaml.Schedule"
  xmlns:res="using:C1.Silverlight.Schedule.Resources"
  FontSize="{ThemeResource ControlContentThemeFontSize}"
  Loaded="EditAppointmentControl_Loaded" x:Name="root" DataContextChanged="OnDataContextChanged"
  mc:Ignorable="d" >
  <UserControl.Resources>
    <ResourceDictionary>
      <ResourceDictionary.ThemeDictionaries>
        <ResourceDictionary x:Key="Default">
          <SolidColorBrush x:Key="RoundButtonForegroundThemeBrush" Color="#FF24AAD8"/>
          <SolidColorBrush x:Key="RoundButtonPointerOverBackgroundThemeBrush" Color="#2124AAD8"/>
        </ResourceDictionary>
        <ResourceDictionary x:Key="Light">
          <SolidColorBrush x:Key="RoundButtonForegroundThemeBrush" Color="#FF24AAD8"/>
          <SolidColorBrush x:Key="RoundButtonPointerOverBackgroundThemeBrush" Color="#2124AAD8"/>
        </ResourceDictionary>
        <ResourceDictionary x:Key="HighContrast">
          <SolidColorBrush x:Key="RoundButtonForegroundThemeBrush" Color="#FFFFFFFF"/>
          <SolidColorBrush x:Key="RoundButtonPointerOverBackgroundThemeBrush" Color="#FF008000"/>
        </ResourceDictionary>
        <ResourceDictionary x:Key="HighContrastBlack">
          <SolidColorBrush x:Key="RoundButtonForegroundThemeBrush" Color="#FFFFFFFF"/>
          <SolidColorBrush x:Key="RoundButtonPointerOverBackgroundThemeBrush" Color="#FF1AEBFF"/>
        </ResourceDictionary>
        <ResourceDictionary x:Key="HighContrastWhite">
          <SolidColorBrush x:Key="RoundButtonForegroundThemeBrush" Color="#FF000000"/>
          <SolidColorBrush x:Key="RoundButtonPointerOverBackgroundThemeBrush" Color="#FF37006E"/>
        </ResourceDictionary>
      </ResourceDictionary.ThemeDictionaries>
      <SolidColorBrush x:Key="highlightBrush" Color="{ThemeResource SystemColorHotlightColor}"/>
      <Style x:Key="roundButtonStyle" TargetType="Button">
        <Setter Property="MinWidth" Value="0"/>
        <Setter Property="Width" Value="42"/>
        <Setter Property="Height" Value="42"/>
        <Setter Property="Margin" Value="0"/>
        <Setter Property="VerticalAlignment" Value="Bottom"/>
        <Setter Property="Foreground" Value="{ThemeResource RoundButtonForegroundThemeBrush}"/>
        <Setter Property="FontFamily" Value="Segoe UI Symbol"/>
        <Setter Property="FontWeight" Value="Normal"/>
        <Setter Property="FontSize" Value="20"/>
        <Setter Property="AutomationProperties.AutomationId" Value="BackButton"/>
        <Setter Property="AutomationProperties.Name" Value="Back"/>
        <Setter Property="AutomationProperties.ItemType" Value="Navigation Button"/>
```

```
            <Setter Property="Template">
               <Setter.Value>
                  <ControlTemplate TargetType="Button">
                     <Grid x:Name="RootGrid">
                        <Grid >
                           <Rectangle x:Name="FocusVisualWhite" Margin="-3" IsHitTestVisible="False"
                                 Stroke="{ThemeResource FocusVisualWhiteStrokeThemeBrush}"
                                 StrokeEndLineCap="Square" StrokeDashArray="1,1" Opacity="0" StrokeDashOffset="1.5"
/>
                           <Rectangle x:Name="FocusVisualBlack" Margin="-3" IsHitTestVisible="False"
                                 Stroke="{ThemeResource FocusVisualBlackStrokeThemeBrush}"
                                 StrokeEndLineCap="Square" StrokeDashArray="1,1" Opacity="0" StrokeDashOffset="0.5"
/>
                           <Ellipse Stroke="{TemplateBinding Foreground}" StrokeThickness="2"
                              VerticalAlignment="Stretch" HorizontalAlignment="Stretch"/>
                           <Ellipse x:Name="BackgroundGlyph" Stroke="{TemplateBinding Foreground}"
StrokeThickness="2"
                              Fill="{TemplateBinding Foreground}" Opacity="0"
                              VerticalAlignment="Stretch" HorizontalAlignment="Stretch"/>
                           <TextBlock x:Name="NormalGlyph" Text="{TemplateBinding Content}" Foreground="
{TemplateBinding Foreground}"
                              VerticalAlignment="Center" HorizontalAlignment="Center" />
                           <TextBlock x:Name="ArrowGlyph" Text="{TemplateBinding Content}" Foreground="
{ThemeResource BackButtonPressedFore
                              VerticalAlignment="Center" HorizontalAlignment="Center" />
                        </Grid>
                        <VisualStateManager.VisualStateGroups>
                           <VisualStateGroup x:Name="CommonStates">
                              <VisualState x:Name="Normal" />
                              <VisualState x:Name="PointerOver">
                                 <Storyboard>
                                    <DoubleAnimation Storyboard.TargetName="BackgroundGlyph"
Storyboard.TargetProperty="Opacity" To="1"
                                    <ObjectAnimationUsingKeyFrames Storyboard.TargetName="BackgroundGlyph"
Storyboard.TargetProperty="Fi
                                       <DiscreteObjectKeyFrame KeyTime="0" Value="{ThemeResource
RoundButtonPointerOverBackgroundThemeB
                                    </ObjectAnimationUsingKeyFrames>
                                 </Storyboard>
                              </VisualState>
                              <VisualState x:Name="Pressed">
                                 <Storyboard>
                                    <DoubleAnimation Storyboard.TargetName="BackgroundGlyph"
Storyboard.TargetProperty="Opacity" To="1"
                                    <DoubleAnimation Storyboard.TargetName="ArrowGlyph"
Storyboard.TargetProperty="Opacity" To="1" Durat
                                    <DoubleAnimation Storyboard.TargetName="NormalGlyph"
Storyboard.TargetProperty="Opacity" To="0" Dura
                                 </Storyboard>
                              </VisualState>
                              <VisualState x:Name="Disabled">
                                 <Storyboard>
```

```
                                <ObjectAnimationUsingKeyFrames Storyboard.TargetName="RootGrid"
Storyboard.TargetProperty="Visibilit
                                    <DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
                                </ObjectAnimationUsingKeyFrames>
                            </Storyboard>
                        </VisualState>
                    </VisualStateGroup>
                    <VisualStateGroup x:Name="FocusStates">
                        <VisualState x:Name="Focused">
                            <Storyboard>
                                <DoubleAnimation Storyboard.TargetName="FocusVisualWhite"
                                        Storyboard.TargetProperty="Opacity" To="1" Duration="0" />
                                <DoubleAnimation Storyboard.TargetName="FocusVisualBlack"
                                        Storyboard.TargetProperty="Opacity" To="1" Duration="0" />
                            </Storyboard>
                        </VisualState>
                        <VisualState x:Name="Unfocused" />
                        <VisualState x:Name="PointerFocused" />
                    </VisualStateGroup>
                </VisualStateManager.VisualStateGroups>
            </Grid>
        </ControlTemplate>
    </Setter.Value>
</Setter>
</Style>
<DataTemplate x:Key="BaseObjectItemPictTextTemplate">
    <Grid Background="Transparent">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="Auto" />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Border Name="imageBorder" BorderBrush="Black" BorderThickness="1"
CompositeMode="MinBlend"
            Height="18" Width="18" Background="{Binding Brush.Brush}"
            HorizontalAlignment="Center" VerticalAlignment="Center" Margin="6,2">
        </Border>
        <TextBlock Grid.Column="1" Text="{Binding Text}" VerticalAlignment="Center"  />
    </Grid>
</DataTemplate>
<res:C1_Schedule_EditAppointment x:Key="C1_Schedule_EditAppointment"/>
</ResourceDictionary>
</UserControl.Resources>

<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" MinWidth="320"/>
        <ColumnDefinition/>
    </Grid.ColumnDefinitions>
    <Rectangle Grid.Column="0" Grid.RowSpan="2">
```

```
        <Rectangle.Fill>
          <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
            <GradientStop Color="#FF5B9E5B" Offset="0.439"/>
            <GradientStop Color="#FF0A630A" Offset="0.703"/>
            <GradientStop Color="#FF368136" Offset="0.123"/>
          </LinearGradientBrush>
        </Rectangle.Fill>
      </Rectangle>
      <Rectangle Grid.Column="1" Grid.RowSpan="2" >
        <Rectangle.Fill>
          <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
            <GradientStop Color="#FF1B4B1B" Offset="0.394"/>
            <GradientStop Color="#FF407848" Offset="0.048"/>
            <GradientStop Color="#FF349B34" Offset="0.697"/>
          </LinearGradientBrush>
        </Rectangle.Fill>
      </Rectangle>
      <Grid Grid.Row="0" Grid.Column="0" VerticalAlignment="Center">
        <Grid.ColumnDefinitions>
          <ColumnDefinition Width="Auto"/>
          <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Button x:Name="backButton" Click="backButton_Click" Grid.Column="0" Content="&#xE112;"
TabIndex="4"
          ToolTipService.ToolTip="{Binding backButton_ToolTip, Source={StaticResource
C1_Schedule_EditAppointment}}"
          Margin="22, 10, 10, 10" VerticalAlignment="Center" Style="{StaticResource roundButtonStyle}"
Background="#FF060606" Foreground="Whit
        <TextBlock Margin="5" Text="{Binding Header, ElementName=root}" TextWrapping="NoWrap"
TextTrimming="CharacterEllipsis"
          Grid.Column="1" FontSize="18" MaxWidth="240" VerticalAlignment="Center"/>
      </Grid>
      <Grid Grid.Row="0" Grid.Column="1" VerticalAlignment="Center">
        <Grid.ColumnDefinitions>
          <ColumnDefinition/>
          <ColumnDefinition Width="Auto"/>
          <ColumnDefinition Width="Auto"/>
        </Grid.ColumnDefinitions>
        <Rectangle Height="2" Fill="{StaticResource highlightBrush}" Grid.ColumnSpan="3" Margin="30,0,0,0"
VerticalAlignment="Bottom" Horizontal
        <TextBox x:Name="subject" TabIndex="0" Padding="2" BorderThickness="0"
          PlaceholderText="{Binding subjectPlaceHolder, Source={StaticResource C1_Schedule_EditAppointment}}"
          Margin="30,0,0,1" Text="{Binding Subject, Mode=TwoWay}" TextChanged="subject_TextChanged"
FontSize="18"
          Grid.Column="0" MaxLength="255" IsColorFontEnabled="True" VerticalAlignment="Center"/>
        <Button x:Name="saveButton" Click="saveButton_Click" Grid.Column="1" Content="&#xE105;"
TabIndex="1"
          ToolTipService.ToolTip="{Binding saveButton_ToolTip, Source={StaticResource
C1_Schedule_EditAppointment}}"
          Margin="10" VerticalAlignment="Center" Style="{StaticResource roundButtonStyle}"
Foreground="White"/>
        <Button x:Name="deleteButton" Grid.Column="2" Content="&#xE106;" TabIndex="2"
```

```
            c1:CommandExtensions.CommandTarget="{Binding
ParentCollection.ParentStorage.ScheduleStorage.Scheduler}"
            c1:CommandExtensions.CommandParameter="{Binding Tag}"
             c1:CommandExtensions.Command="c1sched:C1Scheduler.DeleteAppointmentCommand"
            ToolTipService.ToolTip="{Binding deleteButton_ToolTip1, Source={StaticResource
C1_Schedule_EditAppointment}}"
            Margin="10" VerticalAlignment="Center" Style="{StaticResource roundButtonStyle}"
Foreground="White"/>
        </Grid>
        <StackPanel Grid.Column="0" Grid.Row="1" Margin="20, 10, 10, 10">
          <!-- 日付と時刻 -->
          <TextBlock Margin="5,5,5,0" Text="{Binding startTime, Source={StaticResource
C1_Schedule_EditAppointment}}"/>
          <c1datetime:C1DateTimePicker x:Name="startCalendar" Margin="5" TabIndex="5"
DateTimeChanged="startCalendar_DateTimeChanged"
            Padding="4" TimeFormat="ShortTime" DateFormat="Short" TimeIncrement="0:15"
            FirstDayOfWeek="{Binding
Path=ParentCollection.ParentStorage.ScheduleStorage.Scheduler.CalendarHelper.WeekStart}" />
          <!-- End -->
          <TextBlock Margin="5,5,5,0" Text="{Binding endTime, Source={StaticResource
C1_Schedule_EditAppointment}}"/>
          <c1datetime:C1DateTimePicker x:Name="endCalendar" Margin="5" TabIndex="6"
            Padding="4" DateTimeChanged="endCalendar_DateTimeChanged"
            FirstDayOfWeek="{Binding
Path=ParentCollection.ParentStorage.ScheduleStorage.Scheduler.CalendarHelper.WeekStart}"
            TimeFormat="ShortTime" DateFormat="Short" TimeIncrement="0:15"/>
          <!-- 全日 -->
          <CheckBox x:Name="chkAllDay" Checked="chkAllDay_Checked" Unchecked="chkAllDay_Unchecked"
TabIndex="7"
            Margin="5" IsChecked="{Binding AllDayEvent, Mode=TwoWay}"
            Content="{Binding allDayEvent, Source={StaticResource C1_Schedule_EditAppointment}}"/>
          <!-- ロケーション -->
          <TextBox x:Name="location" Margin="5" IsColorFontEnabled="True" TabIndex="8"
            Header="{Binding location, Source={StaticResource C1_Schedule_EditAppointment}}"
            Text="{Binding Location, Mode=TwoWay}" MaxLength="255"/>
          <!-- 繰り返し -->
          <ComboBox x:Name="howOften" SelectionChanged="howOften_SelectionChanged" TabIndex="9"
            Header="{Binding howOftenComboBoxHeader, Source={StaticResource C1_Schedule_EditAppointment}}"
            MinWidth="140" Margin="5" />
          <StackPanel x:Name="recPanel">
            <CheckBox x:Name="recEndDateSet" Margin="5" Checked="recEndDateSet_Checked"
Unchecked="recEndDateSet_Unchecked" TabIndex="10"
              Content="{Binding recEndDate, Source={StaticResource C1_Schedule_EditAppointment}}" />
            <c1datetime:C1DateTimePicker x:Name="recEndDate" Margin="5" IsEnabled="{Binding IsChecked,
ElementName=recEndDateSet}" TabIndex="11"
              Padding="4" DateFormat="Short" EditMode="Date"
DateTimeChanged="recEndDate_DateTimeChanged"
               FirstDayOfWeek="{Binding
Path=ParentCollection.ParentStorage.ScheduleStorage.Scheduler.CalendarHelper.WeekStart}" />
          </StackPanel>
          <!-- アラーム -->
          <CheckBox x:Name="reminderSet" IsChecked="{Binding ReminderSet, Mode=TwoWay}" Margin="5"
```

```
TabIndex="12"
            Content="{Binding reminderSet, Source={StaticResource C1_Schedule_EditAppointment}}" />
        <c1datetime:C1TimeEditor x:Name="reminderTime" TabIndex="13"
            Format="TimeSpan" Padding="4" Margin="5" IsEnabled="{Binding Path=IsChecked,
ElementName=reminderSet}"
            Increment="00:05" CycleChangesOnBoundaries="False"
            AllowNull="False" Minimum="0:00" Maximum="14.00:00" />
        <!-- その他のプロパティ -->
        <ComboBox x:Name="status" TabIndex="14"
            ItemTemplate="{StaticResource BaseObjectItemPictTextTemplate}"
            ItemsSource="{Binding ParentCollection.ParentStorage.ScheduleStorage.StatusStorage.Statuses}"
            SelectedItem="{Binding Path=BusyStatus, Mode=TwoWay}"
            Header="{Binding showTimeAs, Source={StaticResource C1_Schedule_EditAppointment}}"
            MinWidth="140" Padding="0" Margin="5" />
        <ComboBox Grid.Row="1" TabIndex="15" ItemTemplate="{StaticResource
BaseObjectItemPictTextTemplate}"
            ItemsSource="{Binding ParentCollection.ParentStorage.ScheduleStorage.LabelStorage.Labels}"
            SelectedItem="{Binding Path=Label, Mode=TwoWay}"
            Header="{Binding label, Source={StaticResource C1_Schedule_EditAppointment}}"
            MinWidth="140" Padding="0" Margin="5" />
        <CheckBox IsChecked="{Binding Path=Private, Mode=TwoWay}" Margin="5" TabIndex="16"
            Content="{Binding private_, Source={StaticResource C1_Schedule_EditAppointment}}" />
      </StackPanel>
      <TextBox x:Name="body" Text="{Binding Body, Mode=TwoWay}" Grid.Column="1" Grid.Row="1"
TextWrapping="Wrap" TabIndex="3"
        AcceptsReturn="True" IsSpellCheckEnabled="True" IsColorFontEnabled="True"
ScrollViewer.HorizontalScrollMode="Disabled"
        ScrollViewer.VerticalScrollBarVisibility="Auto" BorderThickness="0" Margin="30,0,0,0" PlaceholderText="Add
text" Foreground="#FFD6FFD0">
        <TextBox.Background>
          <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
            <GradientStop Color="#FF9DEE68" Offset="0.345"/>
            <GradientStop Color="#FF68914D" Offset="0.79"/>
            <GradientStop Color="#FF4C8029" Offset="0.103"/>
          </LinearGradientBrush>
        </TextBox.Background>
      </TextBox>
    </Grid>
</UserControl>
```

▶ C# でコードを書く場合

C#

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
```

```
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using C1.C1Schedule;
using C1.Xaml.Schedule;
using C1.Xaml;
using Windows.UI;

// User Control 項目テンプレートは、http://go.microsoft.com/fwlink/?LinkId=234236 で文書化されています。

namespace SchedulerTest3
{
    public sealed partial class EditAppointmentControl : UserControl
    {
        #region ** fields
        private C1FullscreenDialog _parentWindow = null;
        private Appointment _appointment;
        private C1Scheduler _scheduler;
        private bool _isLoaded = false;

        private TimeSpan _defaultStart;
        private TimeSpan _defaultDuration;

        private bool _updatingRecurrence = false;
        #endregion

        //------------------------------------------------------------------------------
        #region ** initialization
        /// <summary>
        /// <see cref="EditAppointmentControl"/> クラスの新しいインスタンスを作成します。
        /// </summary>
        public EditAppointmentControl()
        {
            this.InitializeComponent();

            // 繰り返しコンボを塗りつぶします
            List<string> recTypes = new List<string>();
            recTypes.Add(C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.recOnce);
            recTypes.Add(C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.recDaily);
            recTypes.Add(C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.recWeekdays);
            recTypes.Add(C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.recMonday);
            recTypes.Add(C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.recTuesday);
            recTypes.Add(C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.recWeekly);
            recTypes.Add(C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.recMonthly);
            recTypes.Add(C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.recYearly);
            howOften.ItemsSource = recTypes;
        }

        private void OnDataContextChanged(FrameworkElement sender, DataContextChangedEventArgs args)
        {
            if (DataContext != null)
            {
                _appointment = DataContext as Appointment;
                _defaultStart = _appointment.AllDayEvent ? TimeSpan.FromHours(8) : _appointment.Start.TimeOfDay;
                _defaultDuration = _appointment.AllDayEvent ? TimeSpan.FromMinutes(30) : _appointment.Duration;
```

```
            if (_appointment != null)
            {
                if (_appointment.ParentCollection != null)
                {
                    _scheduler = _appointment.ParentCollection.ParentStorage.ScheduleStorage.Scheduler;
                    if (_appointment.AllDayEvent)
                    {
                        _defaultStart = _scheduler.CalendarHelper.StartDayTime;
                        _defaultDuration = _scheduler.CalendarHelper.Info.TimeScale;
                    }
                }
                UpdateWindowHeader();
                UpdateRecurrenceState();
                reminderTime.Value = _appointment.ReminderTimeBeforeStart;
                UpdateCalendars();
                if (_appointment.AllDayEvent)
                {
                    startCalendar.EditMode = endCalendar.EditMode =
C1.Xaml.DateTimeEditors.C1DateTimePickerEditMode.Date;
                }
                else
                {
                    startCalendar.EditMode = endCalendar.EditMode =
C1.Xaml.DateTimeEditors.C1DateTimePickerEditMode.DateTime;
                }
            }
        }
    }

    private void EditAppointmentControl_Loaded(object sender, Windows.UI.Xaml.RoutedEventArgs e)
    {
        if (!_isLoaded)
        {
            _parentWindow = (C1FullscreenDialog)VTreeHelper.GetParentOfType(this, typeof(C1FullscreenDialog));
            if (_parentWindow != null && _appointment != null)
            {
                _parentWindow.Unloaded += _parentWindow_Unloaded;
                _isLoaded = true;
            }
        }
        subject.Focus(Windows.UI.Xaml.FocusState.Programmatic);
    }

    void _parentWindow_Unloaded(object sender, RoutedEventArgs e)
    {
        _parentWindow.Unloaded -= _parentWindow_Unloaded;
        DataContext = null;
        _scheduler = null;
        _appointment = null;
        _parentWindow = null;
    }
    #endregion

    //------------------------------------------------------------------------------
```

```
#region ** object model
/// <summary>
/// 現在の DataContext を表す <see cref="Appointment"/> オブジェクトを取得または設定します。
/// </summary>
public Appointment Appointment
{
    get
    {
        return _appointment;
    }
    set
    {
        _appointment = value;
        if (_parentWindow != null)
        {
            _parentWindow.Content =
            _parentWindow.DataContext = value;
        }
        DataContext = value;
        if (_appointment != null)
        {
            UpdateWindowHeader();
            UpdateRecurrenceState();
        }
    }
}

/// <summary>
/// 予定ウィンドウのヘッダーとして使用される <see cref="String"/> 値を取得します。
/// </summary>
public string Header
{
    get { return (string)GetValue(HeaderProperty); }
    private set { SetValue(HeaderProperty, value); }
}

/// <summary>
/// <see cref="Header"/> 依存プロパティを識別します。
/// </summary>
private static readonly DependencyProperty HeaderProperty =
    DependencyProperty.Register("Header", typeof(string),
    typeof(EditAppointmentControl), new PropertyMetadata(string.Empty));
#endregion

//------------------------------------------------------------------------------
#region ** recurrence properties
private void UpdateRecurrenceState()
{
    if (_appointment == null || _updatingRecurrence)
    {
        return;
    }
    if (_appointment.RecurrenceState == RecurrenceStateEnum.NotRecurring)
    {
```

```
            recPanel.Visibility = Windows.UI.Xaml.Visibility.Collapsed;
            howOften.IsEnabled = true;
            howOften.SelectedIndex = 0;
        }
        else
        {
            RecurrencePattern pattern = _appointment.GetRecurrencePattern();
            switch (pattern.RecurrenceType)
            {
                case RecurrenceTypeEnum.Daily:
                    howOften.SelectedIndex = 1;
                    break;
                case RecurrenceTypeEnum.Workdays:
                    howOften.SelectedIndex = 2;
                    break;
                case RecurrenceTypeEnum.Weekly:
                    WeekDaysEnum mask = pattern.DayOfWeekMask;
                    if ((mask & WeekDaysEnum.Monday) == WeekDaysEnum.Monday
                        && (mask & WeekDaysEnum.Wednesday) == WeekDaysEnum.Wednesday
                        && (mask & WeekDaysEnum.Friday) == WeekDaysEnum.Friday
                        && (mask & WeekDaysEnum.Tuesday) != WeekDaysEnum.Tuesday)
                    {
                        howOften.SelectedIndex = 3;
                    }
                    else if ((mask & WeekDaysEnum.Tuesday) == WeekDaysEnum.Tuesday && (mask &
WeekDaysEnum.Thursday) == WeekDaysEnum.Thursday)
                    {
                        howOften.SelectedIndex = 4;
                    }
                    else
                    {
                        howOften.SelectedIndex = 5;
                    }
                    break;
                case RecurrenceTypeEnum.Monthly:
                    howOften.SelectedIndex = 6;
                    break;
                case RecurrenceTypeEnum.Yearly:
                    howOften.SelectedIndex = 7;
                    break;
            }
            if (_appointment.RecurrenceState == RecurrenceStateEnum.Master)
            {
                recPanel.Visibility = Windows.UI.Xaml.Visibility.Visible;
                howOften.IsEnabled = true;
                recEndDate.DateTime = pattern.PatternEndDate;
                recEndDateSet.IsChecked = !pattern.NoEndDate;
            }
            else
            {
                // 発生
                recPanel.Visibility = Windows.UI.Xaml.Visibility.Collapsed;
```

```
            howOften.IsEnabled = false;
        }
    }
}
private void howOften_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (_appointment == null)
    {
        return;
    }
    _updatingRecurrence = true;
    if (howOften.SelectedIndex == 0)
    {
        _appointment.ClearRecurrencePattern();
    }
    else
    {
        RecurrencePattern pattern = _appointment.GetRecurrencePattern();
        switch (howOften.SelectedIndex)
        {
            case 1:
                pattern.RecurrenceType = RecurrenceTypeEnum.Daily;
                break;
            case 2:
                pattern.RecurrenceType = RecurrenceTypeEnum.Workdays;
                break;
            case 3:
                pattern.RecurrenceType = RecurrenceTypeEnum.Weekly;
                pattern.DayOfWeekMask = WeekDaysEnum.Monday | WeekDaysEnum.Wednesday |
WeekDaysEnum.Friday;
                break;
            case 4:
                pattern.RecurrenceType = RecurrenceTypeEnum.Weekly;
                pattern.DayOfWeekMask = WeekDaysEnum.Tuesday | WeekDaysEnum.Thursday;
                break;
            case 5:
                pattern.RecurrenceType = RecurrenceTypeEnum.Weekly;
                pattern.DayOfWeekMask = (WeekDaysEnum)Enum.Parse(typeof(WeekDaysEnum),
Enum.GetName(typeof(DayOfWeek), _appointment.Start.DayOfWeek), true);
                break;
            case 6:
                pattern.RecurrenceType = RecurrenceTypeEnum.Monthly;
                pattern.DayOfMonth = _appointment.Start.Day;
                break;
            case 7:
                pattern.RecurrenceType = RecurrenceTypeEnum.Yearly;
                pattern.DayOfMonth = _appointment.Start.Day;
                pattern.MonthOfYear = _appointment.Start.Month;
                break;
        }
    }
    _updatingRecurrence = false;
```

```
    UpdateRecurrenceState();
    UpdateCalendars();
}
private void recEndDateSet_Unchecked(object sender, RoutedEventArgs e)
{
    if (_appointment != null && _appointment.RecurrenceState == RecurrenceStateEnum.Master)
    {
        RecurrencePattern pattern = _appointment.GetRecurrencePattern();
        pattern.NoEndDate = true;
    }
}

private void recEndDateSet_Checked(object sender, RoutedEventArgs e)
{
    if (_appointment != null && _appointment.RecurrenceState == RecurrenceStateEnum.Master)
    {
        RecurrencePattern pattern = _appointment.GetRecurrencePattern();
        pattern.NoEndDate = false;
        pattern.PatternEndDate = recEndDate.DateTime.Value;
    }
}

private void recEndDate_DateTimeChanged(object sender, object e)
{
    if (_appointment != null && _appointment.RecurrenceState == RecurrenceStateEnum.Master)
    {
        RecurrencePattern pattern = _appointment.GetRecurrencePattern();
        pattern.PatternEndDate = recEndDate.DateTime.Value;
    }
}
#endregion

//------------------------------------------------------------------------------
#region ** times
private void startCalendar_DateTimeChanged(object sender, object e)
{
    if (_appointment != null)
    {
        _appointment.Start = startCalendar.DateTime.Value;
        if (_appointment.RecurrenceState == RecurrenceStateEnum.Master)
        {
            RecurrencePattern pattern = _appointment.GetRecurrencePattern();
            pattern.PatternStartDate = startCalendar.DateTime.Value;
            pattern.StartTime = _appointment.Start;
        }
        UpdateCalendars();
    }
}

private void chkAllDay_Checked(object sender, RoutedEventArgs e)
{
    if (_appointment.RecurrenceState == RecurrenceStateEnum.Master)
    {
        RecurrencePattern pattern = _appointment.GetRecurrencePattern();
        pattern.StartTime = _appointment.Start;
```

```
            pattern.Duration = _appointment.Duration;
        }
        startCalendar.EditMode = endCalendar.EditMode =
C1.Xaml.DateTimeEditors.C1DateTimePickerEditMode.Date;
        UpdateWindowHeader();
        UpdateCalendars();
    }

    private void chkAllDay_Unchecked(object sender, RoutedEventArgs e)
    {
        _appointment.Start = _appointment.Start.Add(_defaultStart);
        _appointment.Duration = _defaultDuration;
        if (_appointment.RecurrenceState == RecurrenceStateEnum.Master)
        {
            RecurrencePattern pattern = _appointment.GetRecurrencePattern();
            pattern.StartTime = _appointment.Start;
            pattern.Duration = _appointment.Duration;
        }
        startCalendar.EditMode = endCalendar.EditMode =
C1.Xaml.DateTimeEditors.C1DateTimePickerEditMode.DateTime;
        UpdateWindowHeader();
        UpdateCalendars();
    }

    private void endCalendar_DateTimeChanged(object sender, object e)
    {
        if (_appointment != null)
        {
            DateTime end = endCalendar.DateTime.Value;
            if (_appointment.AllDayEvent)
            {
                end = end.AddDays(1);
            }
            if (end < Appointment.Start)
            {
                endCalendar.BorderBrush = endCalendar.Foreground = new SolidColorBrush(Colors.Red);
                endCalendar.BorderThickness = new Thickness(2);
                ToolTipService.SetToolTip(endCalendar,
C1.Silverlight.Schedule.Resources.C1_Schedule_Exceptions.StartEndValidationFailed);
                saveButton.IsEnabled = false;
            }
            else
            {
                _appointment.End = end;
                if (!saveButton.IsEnabled)
                {
                    saveButton.IsEnabled = true;
                    endCalendar.ClearValue(Control.ForegroundProperty);
                    endCalendar.ClearValue(Control.BorderBrushProperty);
                    endCalendar.ClearValue(Control.BorderThicknessProperty);
                    endCalendar.ClearValue(ToolTipService.ToolTipProperty);
                }
            }
            if (_appointment.RecurrenceState == RecurrenceStateEnum.Master)
```

```
        {
            RecurrencePattern pattern = _appointment.GetRecurrencePattern();
            pattern.StartTime = _appointment.Start;
            pattern.Duration = _appointment.Duration;
        }
    }
}
private void UpdateCalendars()
{
    if (_appointment.RecurrenceState == RecurrenceStateEnum.Master)
    {
        RecurrencePattern pattern = _appointment.GetRecurrencePattern();
        startCalendar.DateTime = pattern.StartTime;
        DateTime end = pattern.EndTime;
        if (_appointment.AllDayEvent)
        {
            end = end.AddDays(-1);
        }
        endCalendar.DateTime = end;
    }
    else
    {
        startCalendar.DateTime = _appointment.Start;
        DateTime end = _appointment.End;
        if (_appointment.AllDayEvent)
        {
            end = end.AddDays(-1);
        }
        endCalendar.DateTime = end;
    }
    if (!saveButton.IsEnabled)
    {
        saveButton.IsEnabled = true;
        endCalendar.ClearValue(Control.BackgroundProperty);
        endCalendar.ClearValue(Control.ForegroundProperty);
        endCalendar.ClearValue(Control.BorderBrushProperty);
        endCalendar.ClearValue(Control.BorderThicknessProperty);
        endCalendar.ClearValue(ToolTipService.ToolTipProperty);
    }
}
#endregion

//------------------------------------------------------------------------------
#region ** misc props
private void UpdateWindowHeader()
{
    string result;
    string subject = string.Empty;
    bool allDay = false;
    if (_appointment != null)
    {
        subject = _appointment.Subject;
```

```
            allDay = chkAllDay.IsChecked.Value;
        }
        if (String.IsNullOrEmpty(subject))
        {
            subject = C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.Untitled;
        }
        if (allDay)
        {
            result = C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.Event + " - " + subject;
        }
        else
        {
            result = C1.Silverlight.Schedule.Resources.C1_Schedule_EditAppointment.Appointment + " - " + subject;
        }
        Header = result;
    }

    private void subject_TextChanged(object sender, TextChangedEventArgs e)
    {
        subject.GetBindingExpression(TextBox.TextProperty).UpdateSource();
        UpdateWindowHeader();
    }
    #endregion

    //------------------------------------------------------------------------------
    #region ** navigation
    private void saveButton_Click(object sender, RoutedEventArgs e)
    {
        _appointment.ReminderTimeBeforeStart = reminderTime.Value.Value;
        subject.Focus(Windows.UI.Xaml.FocusState.Programmatic);
        location.GetBindingExpression(TextBox.TextProperty).UpdateSource();
        body.GetBindingExpression(TextBox.TextProperty).UpdateSource();
        _parentWindow.DialogResult = MessageBoxResult.OK;
    }

    private void backButton_Click(object sender, RoutedEventArgs e)
    {
        _parentWindow.DialogResult = MessageBoxResult.Cancel;
    }
#pragma warning disable 1591

    protected override void OnKeyDown(Windows.UI.Xaml.Input.KeyRoutedEventArgs e)
    {
        if (e.Key == Windows.System.VirtualKey.Escape)
        {
            backButton_Click(null, null);
        }
        else if (e.Key == Windows.System.VirtualKey.S && KeyboardUtil.Ctrl)
        {
            saveButton_Click(null, null);
        }
        else if (e.Key == Windows.System.VirtualKey.D && KeyboardUtil.Ctrl)
        {
            _scheduler.DeleteAppointment(_appointment.Tag as Appointment);
        }
```

```
            base.OnKeyDown(e);
        }
#pragma warning restore 1591
        #endregion
    }
}
```

アプリケーションに新しい **UserControl** を追加します。

1. アプリケーション名を右クリックし、コンテキストメニューから**[追加]**→**[新しい項目]**を選択します。
2. **[新しい項目の追加]**ダイアログから **User Control** を選択します。
3. 新しい User Control に **EditAppointmentControl** と名前を付けて、**[OK]**をクリックします。アプリケーションにコントロールが追加されます。

EditAppointmentControl.xaml ファイルを開き、ページの上部に XAML マークアップを追加します。このマークアップを編集して、新規予定ダイアログをカスタマイズできます。**EditAppointmentControl.xaml.cs** ファイルにコードを追加します。

カスタムダイアログを作成したので、ユーザーコントロールをロードする **Page** のリソース内で **DataTemplate** を定義します。

XAML

```
<Page.Resources>
    <DataTemplate x:Key="customEditAppointmentTemplate">
        <local:EditAppointmentControl/>
    </DataTemplate>
</Page.Resources>
```

最後に、C1Scheduler コントロールの EditAppointmentTemplate プロパティを設定します。

XAML

```
<Schedule:C1Scheduler x:Name="sched1" EditAppointmentTemplate="{StaticResource
customEditAppointmentTemplate}" />
```

完了しました。**[予定の編集]ダイアログ**を開くと、行ったカスタマイズ内容が表示されます。

コードからの[予定の編集]ダイアログの呼び出し

以下のコードは **DefaultView** サンプルからの抜粋で、**[予定の編集]ダイアログ**をコードから呼び出す例を示しています。

ユーザーがアプリケーションの[追加]ボタンをクリックすると、[予定の編集]ダイアログが表示されます。

C#

```
private void Add_Click(object sender, RoutedEventArgs e)
{
    sched1.NewAppointmentDialog();
}
```

完全なサンプルは、**ComponentOne Samples** フォルダの **Schedule Samples** にあります。

データ連結

予定の連結

Scheduler for UWP では、カスタムデータソースに連結できます。**NestedPropertySetter** を使用して、**AppointmentStorage.DataSource** プロパティ設定することで、予定のプロパティと **AppointmentStorage** クラスの間のマッピングを設定します。

適切な参照と C1Scheduler コントロールが追加されているプロジェクトから始めます。

初期の名前空間宣言と XAML マークアップは、次のサンプルのようになります。

XAML

```
<Page
    x:Class="ScheduleSamples.Samples.BusinessObjectsBinding"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:ScheduleSamples.Samples"
    xmlns:Schedule="using:C1.Xaml.Schedule"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Schedule:C1Scheduler x:Name="sched1"  >
        </Schedule:C1Scheduler>
    </Grid>
</Page>
```

ページを右クリックし、コンテキストメニューから**[コードの表示]**を選択します。コードファイルが開いたら、名前空間宣言をチェックして、次のようになっていることを確認します。

C#

```
using C1.C1Schedule;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.ComponentModel;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using System.Runtime.Serialization;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
```

次に、**MainPage()** コンストラクタを編集して、**AppointmentCollection:** にデモ用の予定を追加します。

C#

```
    public MainPage()
    {
        this.InitializeComponent();
```

```
        sched1.Settings.FirstVisibleTime = System.TimeSpan.FromHours(8);

        AppointmentCollection apps = Resources["_ds"] as AppointmentCollection;
        if (apps != null)
        {
            // デモ用の予定を追加します
            Appointment app = new Appointment();
            app.Subject = "test appointment";
            app.Start = DateTime.Today;
            apps.Add(app);
        }
    }
}
```

次に、カスタム **Appointment** クラスと **PropertyChangedEventHandler** を追加します。

▶ C# でコードを書く場合

C#

```
[DataContract(Name = "Appointment", Namespace = "http://www.componentone.com")]
public class Appointment : INotifyPropertyChanged
{
    public Appointment()
    {
        Id = Guid.NewGuid();
    }
    [DataMember]
    public Guid Id { get; private set; }

    private string _subject = "";
    [DataMember]
    public string Subject
    {
        get { return _subject; }
        set
        {
            if (_subject != value)
            {
                _subject = value;
                OnPropertyChanged("Subject");
            }
        }
    }

    private string _location = "";
    [DataMember]
    public string Location
    {
        get { return _location; }
        set
        {
            if (_location != value)
            {
                _location = value;
                OnPropertyChanged("Location");
```

```
        }
    }
}
private DateTime _start;
[DataMember]
public DateTime Start
{
    get { return _start; }
    set
    {
        if (_start != value)
        {
            _start = value;
            OnPropertyChanged("Start");
        }
    }
}
[DataMember]
public DateTime End
{
    get { return _start.Add(_duration); }
    set
    {
        if (value >= _start)
        {
            Duration = (value.Subtract(_start));
            OnPropertyChanged("End");
        }
    }
}
private TimeSpan _duration;
public TimeSpan Duration
{
    get { return _duration; }
    set
    {
        if (_duration != value)
        {
            _duration = value;
            OnPropertyChanged("Duration");
        }
    }
}
private string _description = "";
[DataMember]
public string Description
{
    get { return _description; }
    set
    {
        if (_description != value)
```

```
        {
            _description = value;
            OnPropertyChanged("Description");
        }
    }
}
private string _properties = "";
[DataMember]
public string Properties
{
    get { return _properties; }
    set
    {
        if (_properties != value)
        {
            _properties = value;
            OnPropertyChanged("Properties");
        }
    }
public event PropertyChangedEventHandler PropertyChanged;
        protected void OnPropertyChanged(string propertyName)
        {
            if (PropertyChanged != null)
                PropertyChanged(this, new PropertyChangedEventArgs(propertyName));
        }
    }
```

ObservableCollection を継承する **AppointmentCollection** クラスを追加します。

C#

```
public class AppointmentCollection : ObservableCollection<Appointment>
{
    public AppointmentCollection()
    {
    }
}
```

次に、**MainPage.xaml** ページでカスタムデータソースのインスタンスを作成します。ローカルリソースの Key 値に、先にコードで定義した名前を設定します。

XAML

```
<Page.Resources>
    <local:AppointmentCollection x:Key="_ds"/>
</Page.Resources>
```

NestedPropertySetter を使用して、**Appointment** クラスで設定された予定のプロパティと AppointmentStorage クラスの間のマッピングを設定します。

XAML

```
<c1:C1Scheduler x:Name="sched1" >
    <!-- AppointmentStorage をビジネスオブジェクトのコレクションにマップします -->
```

```
    <c1:NestedPropertySetter
        PropertyName="DataStorage.AppointmentStorage.Mappings.AppointmentProperties.MappingName"
        Value="Properties"/>
    <c1:NestedPropertySetter
        PropertyName="DataStorage.AppointmentStorage.Mappings.Body.MappingName"
        Value="Description"/>
    <c1:NestedPropertySetter
        PropertyName="DataStorage.AppointmentStorage.Mappings.End.MappingName"
        Value="End"/>
    <c1:NestedPropertySetter
        PropertyName="DataStorage.AppointmentStorage.Mappings.IdMapping.MappingName"
        Value="Id"/>
    <c1:NestedPropertySetter
        PropertyName="DataStorage.AppointmentStorage.Mappings.Location.MappingName"
        Value="Location"/>
    <c1:NestedPropertySetter
        PropertyName="DataStorage.AppointmentStorage.Mappings.Start.MappingName"
        Value="Start"/>
    <c1:NestedPropertySetter
        PropertyName="DataStorage.AppointmentStorage.Mappings.Subject.MappingName"
        Value="Subject"/>
    <c1:NestedPropertySetter PropertyName="DataStorage.AppointmentStorage.DataSource"
        Value="{Binding Mode=TwoWay, Source={StaticResource _ds}}" />
</c1:C1Scheduler>
```

アプリケーションを実行すると、現在の日にマークが付いたテスト予定が表示されます。

ラベルの連結

独自のラベルコレクションを作成し、C1Scheduler コントロールのラベルデータストレージにそれらを連結できます。たとえば、カスタム Label クラスを定義するか、**C1.C1Schedule.Label** クラスを使用します。

C#

```
public class MyLabel
{
    public string Id { get; set; }
    public string Text { get; set; }
    public string Color { get; set; }
}
```

Id プロパティや Index プロパティは必須でありませんが、データを保存する目的では Id（Guid/string）または Index（int）のいずれかの識別子を各ラベルに入れる必要があります。これにより、アプリケーションを次回実行するまで予定のラベル情報を保存できます。

ラベルの色

ラベルの色を指定するために、**C1Scheduler** は「255,255,255,255」などの文字列値を使用します。これは、アルファ、赤、青、緑の各バイトチャンネルを表します。これにより、使用可能なすべての色に対して、シリアライズと最適なストレージ容量を利用できます。

コードでラベルのコレクションを作成し、それを **LabelStorage.DataSource** プロパティに連結します。また、**ID**、**Text**、**Color** の各プロパティに適切なマッピングを設定する必要があります。

▶ C# でコードを書く場合

C#

```
// ラベルのリストを作成します
 List<MyLabel> Labels = new List<MyLabel>();
 Labels.Add(new MyLabel { Text = "Lime", Color = "255,164,196,0" });
 Labels.Add(new MyLabel { Text = "Green", Color = "255,96,169,23" });
 Labels.Add(new MyLabel { Text = "Teal", Color = "255,0,171,169" });
 Labels.Add(new MyLabel { Text = "Cyan", Color = "255,27,161,226" });
 Labels.Add(new MyLabel { Text = "Purple", Color = "255,170,0,255" });
 Labels.Add(new MyLabel { Text = "Pink", Color = "255,244,141,208" });
 Labels.Add(new MyLabel { Text = "Red", Color = "255,229,20,0" });
 Labels.Add(new MyLabel { Text = "Mango", Color = "255,240,150,9" });
 Labels.Add(new MyLabel { Text = "Yellow", Color = "255,227,200,0" });
 Labels.Add(new MyLabel { Text = "Olive", Color = "255,109,135,100" });
 Labels.Add(new MyLabel { Text = "Mauve", Color = "255,118,96,138" });


// ラベルデータストレージとマッピングを設定します
 c1Scheduler.DataStorage.LabelStorage.DataSource = Labels;
 c1Scheduler.DataStorage.LabelStorage.Mappings.TextMapping.MappingName = "Text";
 c1Scheduler.DataStorage.LabelStorage.Mappings.ColorMapping.MappingName = "Color";
 c1Scheduler.DataStorage.LabelStorage.Mappings.IdMapping.MappingName = "Id";
 Alternatively, you can also data bind in XAML. This snippet assumes you have a property named “Labels” in view
model declared as a resource elsewhere in your project.
 <c1sched:C1Scheduler x:Name="c1Scheduler" AppointmentForeground="Black">
    <!-- ラベルストレージをマップします -->
    <c1sched:NestedPropertySetter PropertyName="DataStorage.LabelStorage.DataSource"
                          Value="{Binding Path = Labels, Source={StaticResource mainViewModel}}" />
    <c1sched:NestedPropertySetter
PropertyName="DataStorage.LabelStorage.Mappings.TextMapping.MappingName"
                          Value="Text"/>
    <c1sched:NestedPropertySetter
PropertyName="DataStorage.LabelStorage.Mappings.ColorMapping.MappingName"
                          Value="Color"/>
    <c1sched:NestedPropertySetter
PropertyName="DataStorage.LabelStorage.Mappings.IdMapping.MappingName"
                          Value="Id"/>
 </c1sched:C1Scheduler>
```

アラーム

デフォルトでは、C1Scheduler コントロールはアラームを表示しません。アラームの表示は開発者がコードでカスタマイズする必要がありますが、C1Scheduler は、いくつかのアラームを簡単に実装できるように作成されています。

- **Toast 通知** - アプリケーションが実行されていない場合でも、アラームが機能するように設定します。
- **組み込みのダイアログ** - アプリケーションの実行中にのみアラームがポップアップされます。

Toast 通知

組み込みのアラームダイアログボックスの代わりに Toast 通知を使用すると、アプリケーションが実行されているかどうかに関

係なく、ユーザーは通知を受け取ることができます。エンドユーザーは、その通知をクリックしてアプリケーションを起動することもできます。Toast 通知プロパティの完全な実装は、次の **ComponentOne Samples** フォルダの **Save Local Data** サンプル内にあります。

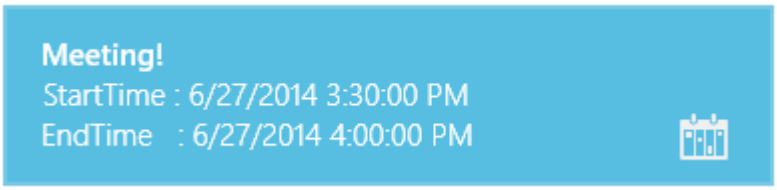
C:\Users\YourUserName\Documents\ComponentOne Samples\UWP\C1.Xaml.Schedule\CS\ScheduleSamples

Toast 通知の設定は簡単です。**MainPage()** コンストラクタ内で次のコードを使用するだけです。これは、try-catch ステートメント内に入っていることに注意してください。

C#

```
try
{
   // C1Schedule に Toast 通知を設定して、埋め込みのアラームダイアログボックスの代わりに Toast 通知を表示します
   sched1.ToastNotifier = Windows.UI.Notifications.ToastNotificationManager.CreateToastNotifier();
   // エンドユーザーが何らかの理由で通知を無効にした場合、通知は表示されません
   // アプリケーションがシミュレータで実行されている場合も、Toast 通知は無効です
}
catch
{
   // アプリケーションのマニフェストが Toast 通知を許可しない場合、上のコードは失敗します
}
```

Toast 通知は次の図のように表示されます。これがトリガされると、エンドユーザーの画面の右上に通知が表示されます。



組み込みのダイアログ

ShowReminderDialog プロパティを true に設定して、組み込みのアラームダイアログボックスを有効にすることができます。アラームが発生すると、デフォルトのアラームが表示されます。

C#

```
sched1.Settings.ShowReminderDialog = true;
```

上のマークアップにより、次の画像のようなアラームダイアログボックスが表示されます。

ビュー

C1Scheduler コントロールでは、**月**、**週**、**週間勤務日**、**日**の4つのビューを選択できます。初期カレンダービューの変更は、1つのプロパティ ViewType を設定するだけで簡単に行うことができます。デフォルトでは、**月**ビューが表示されます。

次の XAML マークアップサンプルは、**ViewType** プロパティを示します。

月ビュー

```
<c1:C1Scheduler Name="sched1" ViewType="Month"></c1:C1Scheduler>
```

週ビュー

```
<c1:C1Scheduler Name="sched1" ViewType="Week"></c1:C1Scheduler>
```

週間勤務日ビュー

```
<c1:C1Scheduler Name="sched1" ViewType="WorkingWeek"></c1:C1Scheduler>
```

日ビュー

```
<c1:C1Scheduler Name="sched1" ViewType="Day"></c1:C1Scheduler>
```

1行のコードで ViewType を設定することもできます。

月ビュー

```
sched1.ViewType = ViewType.Month;
```

週ビュー

```
sched1.ViewType = ViewType.Week;
```

週間勤務日ビュー

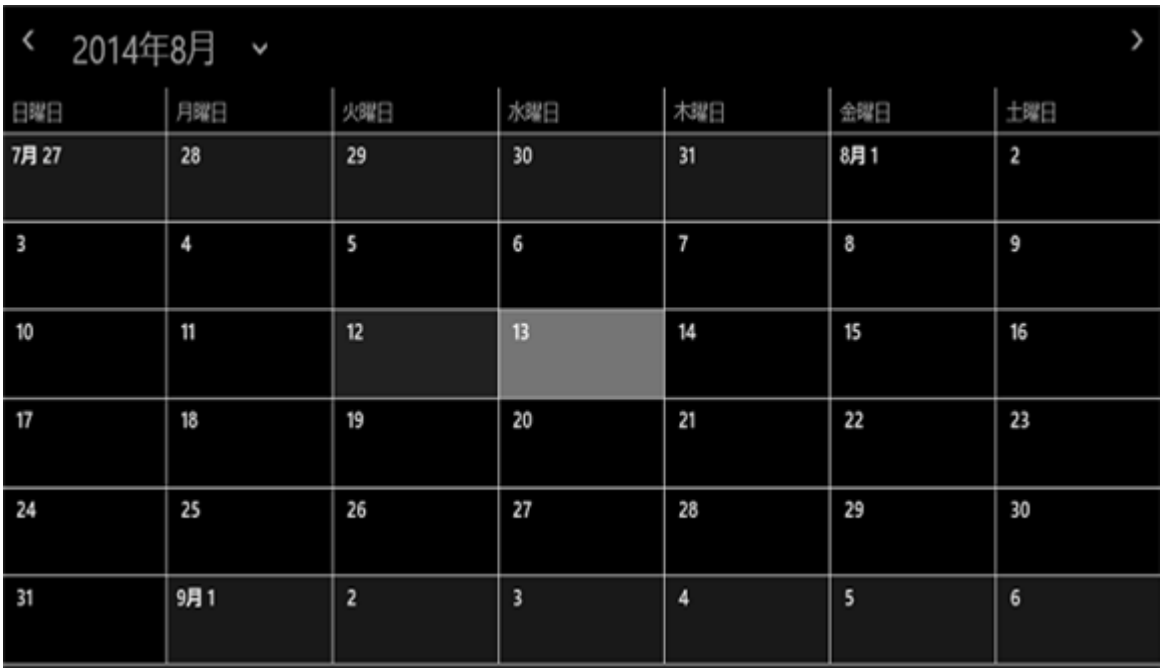
```
sched1.ViewType = ViewType.WorkingWeek;
```

日ビュー

```
sched1.ViewType = ViewType.Day;
```

ヘッダーテキストを選択すると、そのビューが表示されます。

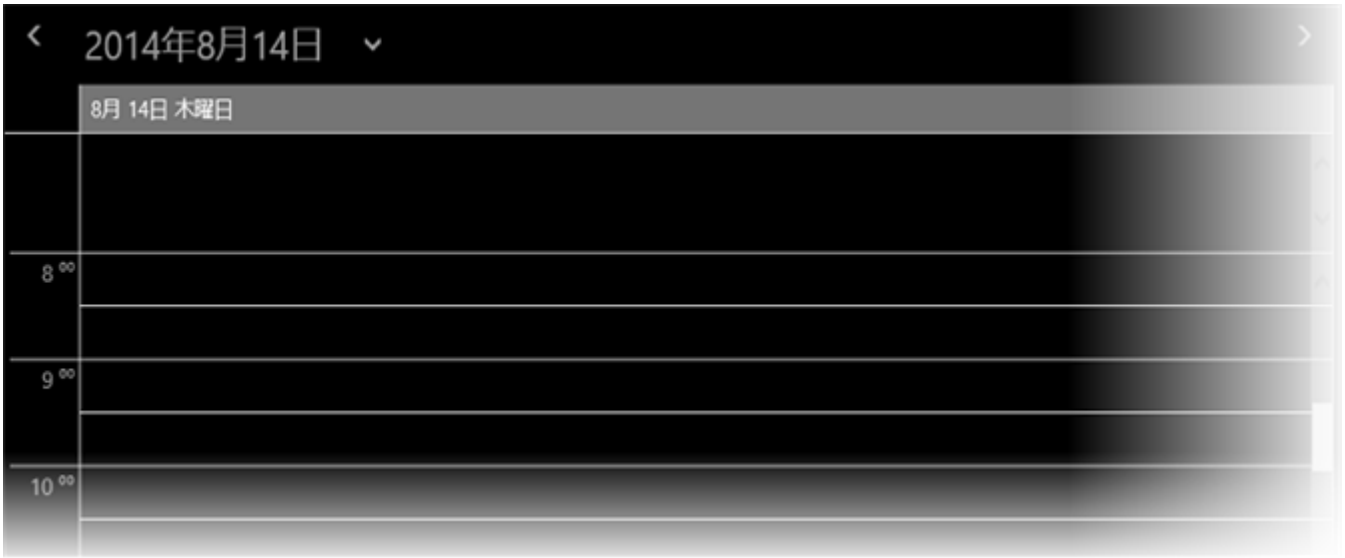
月ビュー



週ビュー

週間勤務日ビュー

日ビュー



週間勤務日の設定

CalendarHelper クラスと **WorkDays** プロパティを使用して、週間勤務日ビューを構成する日を指定できます。このプロパティは、コードで設定する必要があります。マークアップでは設定できません。

次のコードを使用して、WorkDays リストを作成します。

C#

```
C1.C1Schedule.WorkDays workdays = new C1.C1Schedule.WorkDays();
workdays.Add(System.DayOfWeek.Tuesday);
workdays.Add(System.DayOfWeek.Wednesday);
workdays.Add(System.DayOfWeek.Thursday);
workdays.Add(System.DayOfWeek.Friday);
workdays.Add(System.DayOfWeek.Saturday);
sched1.CalendarHelper.WorkDays = workdays;
```

アプリケーションを実行すると、週間勤務日として火曜日から土曜日までが表示されます。

休日の追加

CalendarHelper クラスを使用すると、C1Scheduler コントロールに簡単に休日を追加できます。数行のコードを使用して、特定の日を休日に指定できます。休日として指定した日は、**ViewType** プロパティが **WorkWeek** に設定されている C1Scheduler には表示されません。

次のコードを使用して、カレンダーに休日を設定します。以下のコードは、クリスマス（12 月 25 日）を休日として設定します。

C#

```
sched1.CalendarHelper.Holidays.Add(new DateTime(2014, 12, 25));
```

クリスマスの週の週間勤務日ビューを次に示します。

インポートとエクスポート

予定データを保存して、次回アプリケーションを実行したときにロードできます。C1Scheduler コントロールは、XML と iCal（iCalendar）の2つの使い慣れた形式で、簡便な**インポート**および**エクスポート**方法を提供します。

データのエクスポート

エンドユーザーが C1Schedule コントロールからデータをエクスポートする方法は2つあります。一方では、ユーザーがファイルの保存先を選択できます。もう一方では、エクスポートをサイレントで行うように設定できます。

エクスポートするファイルを **Click** イベントで処理し、ユーザーが C1Scheduler ファイルの保存先を選択できるようにする場合は、次のコードを使用します。

C#

```
// [ファイルの保存]（システム定義）ダイアログを開き、顧客がデータの保存先を選択できるようにしてから、すべての予定をそこにエクスポートします
C1Scheduler.ExportCommand.Execute(null, sched1);
```

エクスポートするデータをサイレントで処理することもできます。

アプリケーションフォルダ内のローカル XML または iCal ファイルにデータをエクスポートするには、最初にページの上部に **System.IO** 名前空間を追加します。

C#

```
using System.IO;
```

次に、**保存ボタン**のクリック、**Page Navigated From**、または **Unloaded** イベントに次のコードを置きます。

C#

```
// アプリケーションのローカルフォルダに XML をエクスポートします
var folder = Windows.Storage.ApplicationData.Current.LocalFolder;
var file = await folder.CreateFileAsync("appointments.xml", Windows.Storage.CreationCollisionOption.OpenIfExists);

IRandomAccessStream stream = await file.OpenAsync(Windows.Storage.FileAccessMode.ReadWrite);
await sched1.DataStorage.ExportAsync(stream.AsStreamForWrite(), C1.C1Schedule.FileFormatEnum.XML);

await stream.FlushAsync();
stream.Dispose();
```

データのインポート

アプリケーションにデータをロードする方法は2つあります。ユーザーがインポート対象を選択できるようにするか、インポートをサイレントで処理できます。

インポートするデータを処理し、アプリケーションにロードするファイルをユーザーが選択できるようにするには、次のコードを使用します。

C#

```
// [ファイルを開く](システム定義)ダイアログを開き、エンドユーザーがインポートするファイルを選択できるようにします
C1Scheduler.ImportCommand.Execute(null, sched1);
```

インポートするデータをサイレントで処理するには、次のコードを使用します。

アプリケーションフォルダのローカル XML または iCal ファイルからデータをインポートするには、**ロードボタン**のクリック、**Page Navigated To**、または **Loaded** イベントに次のコードを置きます。

C#

```
// アプリケーションのローカルフォルダから XML をインポートします
var folder = Windows.Storage.ApplicationData.Current.LocalFolder;
try
{
    var file = await folder.GetFileAsync("appointments.xml");
    IRandomAccessStream stream = await file.OpenAsync(Windows.Storage.FileAccessMode.Read);
    await sched1.DataStorage.ImportAsync(stream.AsStreamForRead(), C1.C1Schedule.FileFormatEnum.XML);
    stream.Dispose();
}
catch { }
```