XAML Do-It-Yourself シリーズ

第 2 回 レイアウト

Microsoft®

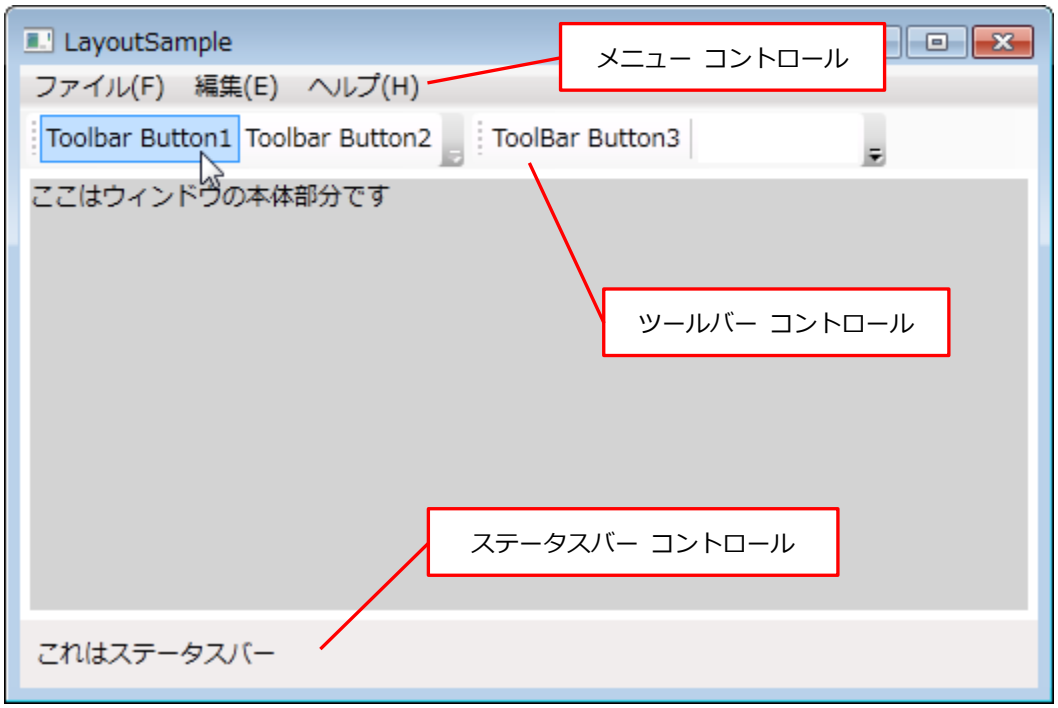
XAMLDo-It-Yourself
第 2 回. レイアウト

今回は、XAML でウィンドウにパネルやボタン、メニューなどの基本的なコントロールを配置する方法を学びます。

ここでは、次のことを学習します。

- XAML におけるレイアウトの基本
- 主要なパネルとその記述方法
- XAML (WPF) で利用可能なコントロール
- 典型的なコントロールの記述方法
- 標準的なウィンドウのレイアウト例

■今回作成する XAML プログラム



今回は、XAML のレイアウト機能を使っていくつかのコントロールを配置し、一般的な Windows アプリケーションのウィンドウを作成します。このウィンドウの内部には、メニューやツール バー、ステータスバーの各コントロールが配置されています。

■レイアウトの基本

XAML を用いた WPF アプリケーションにおけるボタンやテキストボックスといったコントロールの配置は、これまでの Windows フォーム アプリケーションのそれとはやや異なります。一言でいえば、こ

れまでの Windows フォーム アプリケーションでの手法に、ASP.NET (HTML) でのコントロールの配置方法がミックスされています。

また、前回説明したように、XAML でのコントロールの記述は階層的に行います。ルート要素である <Window> 要素の下に必要なコントロールを記述してユーザー インターフェイスを構築していきますが、ルート要素配下に配置できる要素は 1 つだけという決まりがあります。そのためルート要素には後述する Grid や StackPanel といったパネルを記述し、パネルの中にさらに必要なコントロールを記述していきます。パネル内には複数の要素 (コントロール) を配置できます。

●デバイス独立ピクセル

また WPF では、座標の位置を記述する際のデフォルトの単位はピクセル値 (px) です。ピクセルのほかに、インチ (inch)、センチ メートル (cm)、ポイント (pt、1pt = 96 / 72px) が使用できます。WPF のピクセル値は、特定のデバイスに依存しない「デバイス独立ピクセル」値です。具体的には、WPF のピクセルは、96 DPI (Dot Per Pixel。96 DPI は、1 インチに 96 個の点が描画できるということ) を基準にしており、ディスプレイの設定が 96 DPI でない場合は、96 DPI で表示する場合と同じ大きさに表示されるように、自動的にスケール処理されます。このため、表示するデバイスの解像度 (DPI) によらず、表示結果が同じサイズになります。

■パネルの種類と違い

いま述べたように、パネルはコントロールを配置するうえで重要な役割を果たします。以下に、XAML で利用できるパネルについて、主要なものを紹介します。

●Canvas

Canvas は最も単純なパネルです。Canvas の子要素として配置するコントロールの位置は、子要素内で Canvas.Top と Canvas.Left といった属性を使って指定します。

```
<Canvas>
  <Button
    Canvas.Left="20"
    Canvas.Top="30"
    Width="100"
    Height="40"
  >Hello</Button>
</Canvas>
```

●StackPanel

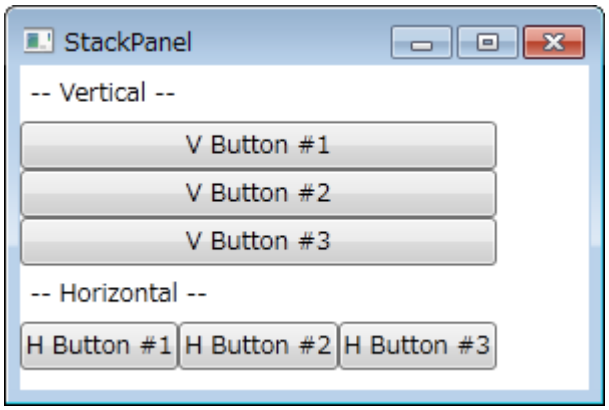
StackPanel は子要素を縦または横に並べて表示します。並べる方向は、Orientation 属性で指定します。

```
<Window x:Class="WpfApplication2.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
```

```
        Title="StackPanel" Height="200" Width="300">
  <Grid>
    <!-- StackPanel を縦に並べる -->
    <StackPanel Orientation="Vertical" HorizontalAlignment="Left" VerticalAlignment="Top">
      <Label>-- Vertical --</Label>
      <StackPanel Orientation="Vertical">
        <Button>V Button #1</Button>
        <Button>V Button #2</Button>
        <Button>V Button #3</Button>
      </StackPanel>
      <Label>-- Horizontal --</Label>
      <StackPanel Orientation="Horizontal">
        <Button>H Button #1</Button>
        <Button>H Button #2</Button>
        <Button>H Button #3</Button>
      </StackPanel>
    </StackPanel>
  </Grid>
</Window>
```

リスト 2-1 StackPanel の例

実行すると、ウィンドウは次のようになります。



●WrapPanel

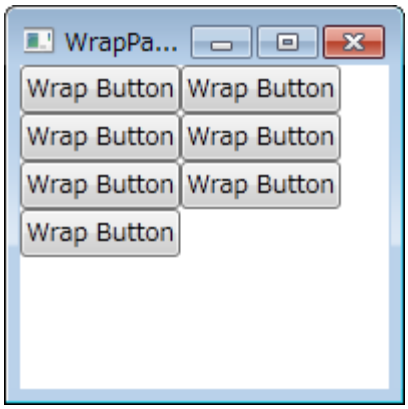
WrapPanel は左上から右方向に順番にコントロールを表示します。領域内に表示しきれないコントロールは自動的に改行され、次の行に表示されます。従って WrapPanel では、ウィンドウのサイズ変更に応じてコントロールの表示位置が変化します。

```
<Window x:Class="_02WpfApplication.Window4"
```

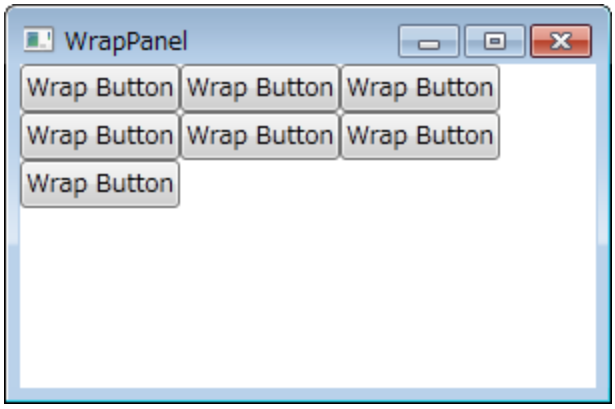
```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window4" Height="300" Width="200">
    <Grid>
    <WrapPanel>
      <Button Width="80" >Wrap Button #1</Button>
      <Button Width="80" >Wrap Button #2</Button>
      <Button Width="80" >Wrap Button #3</Button>
      <Button Width="80" >Wrap Button #4</Button>
      <Button Width="80" >Wrap Button #5</Button>
      <Button Width="80" >Wrap Button #6</Button>
      <Button Width="80" >Wrap Button #7</Button>
    </WrapPanel>
  </Grid>
</Window>
```

リスト 2-2 WrapPanel の例

ウィンドウの表示は次のようになります。



ここで、ウィンドウの横幅を少し広げてみます。



するとこのように、ボタンの位置が自動的に変わります。ウィンドウ サイズをユーザーが自由に変更できるようにした場合に、ウィンドウ サイズに応じてレイアウトを自動的に変化させたい場合に使用します。

●DockPanel

DockPanel は、DockPanel の子要素が DockPanel の上下左右のいずれかにドッキングするパネルです。子要素では、DockPanel.Doc 属性によりドッキングする位置を指定します。

```
<Window x:Class="WpfApplication2.DockPanelWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="DockPanel" Height="300" Width="300">
  <DockPanel>
    <TextBlock DockPanel.Dock="Top" Height="40" Background="Lightgray">
      Top Area (メニュー)
    </TextBlock>
    <TextBlock DockPanel.Dock="Bottom" Height="30" Background="LightGray">
      Bottom Area (ステータスバー)
    </TextBlock>
    <TextBlock DockPanel.Dock="Left" Width="120" Background="Yellow">
      Left Area (ツリー)
    </TextBlock>
    <TextBlock DockPanel.Dock="Left" Background="Pink">
      Right Area(リスト)
    </TextBlock>
  </DockPanel>
</Window>
```

リスト 2-3 DockPanel の例

ウィンドウの表示は次のようになります。

Web のコンテンツ ページでは一般的な、パネル型表示を行う場合に利用できます。

●Grid

Grid は子要素を表のように配置するパネルです。Grid では、Grid の行数と列数に合わせて、行と列の情報を事前に定義します。具体的には、Grid.ColumnDefinitions で列情報を定義し、Grid.RowDefinitions で行情報を定義します。行、列それぞれに幅と高さの指定が可能ですが、自動設定するときは "Auto" を、「ほかのセルが定義した部分を除くすべて」を設定するときは "*" を指定します。

次の例では、最初の列の幅を 100 ピクセルに定義し、2 番目の列の幅はウィンドウ幅の残りすべてを使うように定義しています。また、先頭行の高さは「自動」でセットされるようにし、2 番目の行は 200 ピクセルを確保するように定義しています。

```
<!-- 列の設定 -->
<Grid.ColumnDefinitions>
  <ColumnDefinition Width="100"/>
  <ColumnDefinition Width="*"/>
</Grid.ColumnDefinitions>
<!-- 行の設定 -->
<Grid.RowDefinitions>
  <RowDefinition Height="Auto"/>
  <RowDefinition Height="200"/>
</Grid.RowDefinitions>
```

行数と列数の定義を行った後、実際のセルの中身を記述します。どのセルに対する記述なのかは、

Grid.Column と Grid.Row 属性で指定します。例えば 0 行、0 列目のセルを指定するなら次のようします。

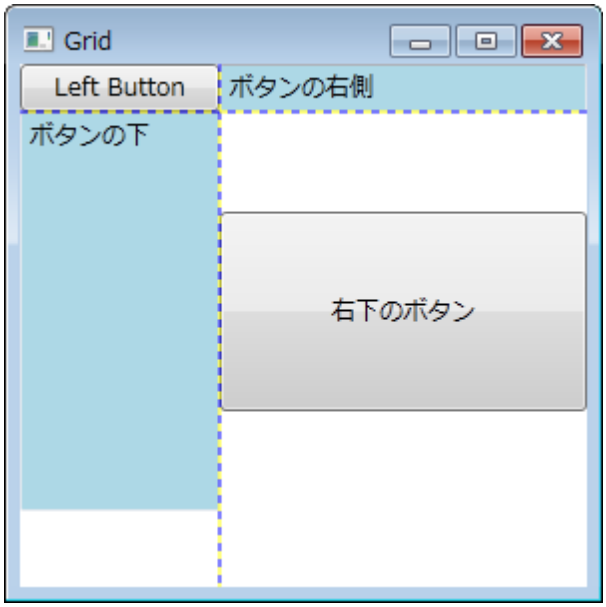
Grid.Column="0" Grid.Row="0"

以下は、Grid を利用して 2 行 2 列のセル領域を作成し、左上と右下のセルには Button を、右上と左下のセルには TextBox をそれぞれ配置した例です。

```
<Window x:Class="WpfApplication2.GridWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Grid" Height="300" Width="300">
  <Grid ShowGridLines="True">
    <!-- 桁の設定 -->
    <Grid.ColumnDefinitions>
      <ColumnDefinition Width="100"/>
      <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <!-- 行の設定 -->
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <RowDefinition Height="200"/>
    </Grid.RowDefinitions>
    <!--各セル -->
    <Button Grid.Column="0" Grid.Row="0">Left Button</Button>
    <TextBox Grid.Column="1" Grid.Row="0" Background="Lightblue">ボタンの右側</TextBox>
    <TextBox Grid.Column="0" Grid.Row="1" Background="Lightblue">ボタンの下</TextBox>
    <Button Grid.Column="1" Grid.Row="1" Height="100">右下のボタン</Button>
  </Grid>
</Window>
```

リスト 2-4 Grid の例

ウィンドウの表示は次のようになります。



■コントロール

パネルを用いたレイアウトの基本について説明しましたので、次にユーザー インターフェイスの中心となるコントロールについて説明します。

XAML (WPF) で利用できるコントロールの種類は、Windows フォームで利用可能なコントロールとさほど変わりありません。主なコントロールを以下の表にまとめました。コントロール名は XAML で記述する要素名に対応しますから、ここで示しているコントロール名を要素名として XAML で記述します。

コントロール名	機能
Border	別の要素の周囲に境界線、背景、またはその両方を描画します
Button	ボタン
CheckBox	チェック ボックス
ComboBox	コンボ ボックス
DataGrid	データを表形式で表示する
Image	画像を描画します
Label	文字列を表示します
ListBox	リスト ボックス
RadioButton	ラジオ ボタン
TabControl	タブ表示
GridSplitter	グリッドを分割してサイズの変更を可能にします
GroupBox	グループ ボックス
ListView	リスト ビュー
MediaElement	オーディオとビデオまたはそのいずれかを含むコントロールを表します

Menu	メニュー
PasswordBox	パスワード (マスク) 入力を行うテキスト ボックス
ProgressBar	プログレス バー
Rectangle	矩形を描画します。そのほかの 2D グラフィックス描画として、Ellipse (だ円)、Line (直線)、Path (連続する直線と曲線)、Polygon (多角形)、Polyline (連続する直線)などがあります
RichTextBox	書式付きのテキスト入力ボックス
ScrollBar	スクロール バー
Separator	メニューなどの項目を区切るために使用します
Slider	スライダー
StatusBar	ステータス バー
TextBox	テキスト入力ボックス
ToolBar	ツール バー
ToolBarTray	ToolBar を配置するコンテナー
TreeView	ツリー ビュー
WebBrowser	ブラウザーのように HTML ドキュメントをホストする
WindowsFormsHost	Windows フォーム コントロールを WPF ページ上にホストできるようにします

表 2-1 XAML (WPF) で利用できる主なコントロール

すでに、Button や TextBox コントロールを使ったサンプルは紹介しました。以降では、典型的なコントロールの記述例として、ListBox、Menu、ToolBar を紹介します。

●ListBox (リスト・ボックス)

ListBox では、一覧表示する項目を事前に用意しておく必要があります。これらは <ListBoxItem> タグを使って記述します。XAML では、ListBox の項目に図形や画像を指定することも可能です。例えば次は、文字列の項目「項目＃1」と「項目＃2」、ビットマップの項目を含むリスト ボックスの例です。

```
<Window x:Class="WpfApplication2.ListBoxWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="ListBox" Height="300" Width="300">
  <Grid>
    <ListBox Width="240" Height="150">
      <ListBoxItem IsSelected="True">項目＃1</ListBoxItem>
      <ListBoxItem >項目＃2</ListBoxItem>
      <ListBoxItem >
        <Image Width="200" Source="/WpfApplication2;component/Images/Penguins.jpg"/>
      </ListBoxItem>
```

```
    </ListBox>
  </Grid>
</Window>
```

表示は次のようになります。

●Menu (メニュー)

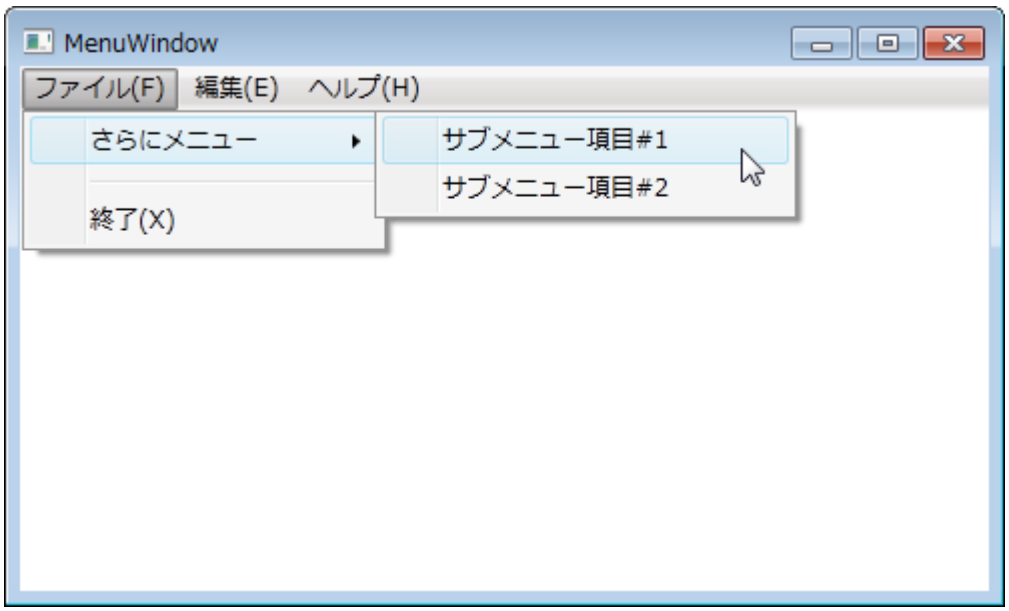
メニューは、<Menu> 要素と <MenuItem> 要素を使って記述します。<MenuItem> 要素をネストすることで、階層的なメニューを作ることも可能です。

なお、次のリストで示しているように、ショートカット キーを示す文字は “&” ではなく “_” (アンダースコア) になっています。これは、XML のルールでは "&" を & と記述する必要があるためで、XAML では、より簡単に記述できる“_”に変更されています。また、メニュー項目の間に区切り線 (セパレーター) を入れたい場合は <Separator> 要素を記述します。

```
<Window x:Class="WpfApplication2.MenuWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="MenuWindow" Height="300" Width="500">
  <DockPanel>
    <Menu DockPanel.Dock="Top" Height="Auto">
      <MenuItem Header="ファイル(_F)">
        <MenuItem Header="さらにメニュー">
          <MenuItem Header="サブメニュー項目#1"/>
          <MenuItem Header="サブメニュー項目#2"/>
```

```
        </MenuItem>
        <Separator/>
        <MenuItem Header="終了(_X)"/>
      </MenuItem>
      <MenuItem Header="編集(_E)">
        <MenuItem Header="コピー"/>
      </MenuItem>
      <MenuItem Header="ヘルプ(_H)">
        <MenuItem Header="○○○について" />
      </MenuItem>
    </Menu>
    <Grid />
  </DockPanel>
</Window>
```

表示は次のようになります。



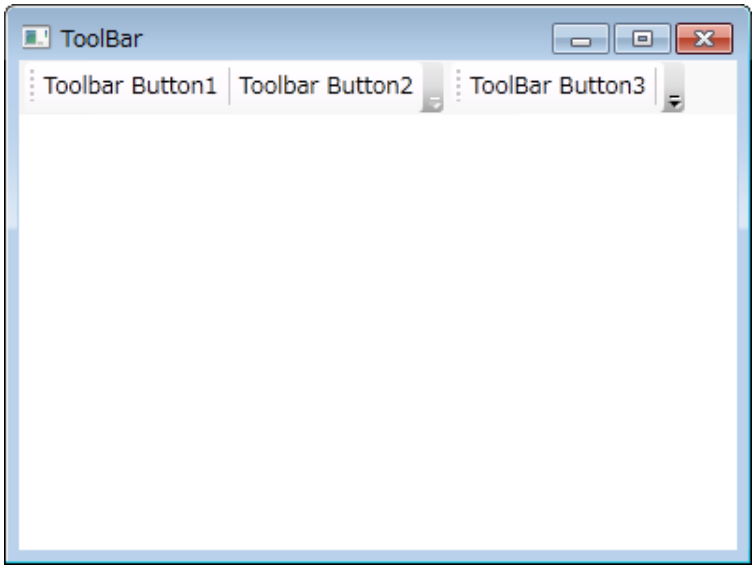
●ToolBar (ツール バー)

ツール バーは、まず <ToolBarTray> 要素でツールバーの領域を確保しておき、そこに <ToolBar> 要素を使って配置します。1 つの<ToolBar> 要素が、1 つのツールバーのまとまりを示します。次のリストは、2 つのツール バーを作成し、第 1 のツール バーには 2 つのボタンを、第 2 のツール バーにはボタンとテキスト ボックスをそれぞれ配置した例です。

```
<Window x:Class="WpfApplication2.ToolBarWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="ToolBar" Height="300" Width="400">
  <Grid>
    <ToolBarTray VerticalAlignment="Top">
      <ToolBar Name="toolBar1" Band="0" BandIndex="0">
        <Button>Toolbar Button1</Button>
        <Separator/>
        <Button>Toolbar Button2</Button>
      </ToolBar>
      <ToolBar Name="toolBar2" Band="0" BandIndex="1">
        <Button>ToolBar Button3</Button>
        <Separator/>
        <TextBox Name="textBox1" Width="80" />
      </ToolBar>
    </ToolBarTray>
  </Grid>
</Window>
```

表示は次のようになります。



■標準的なウィンドウのレイアウト

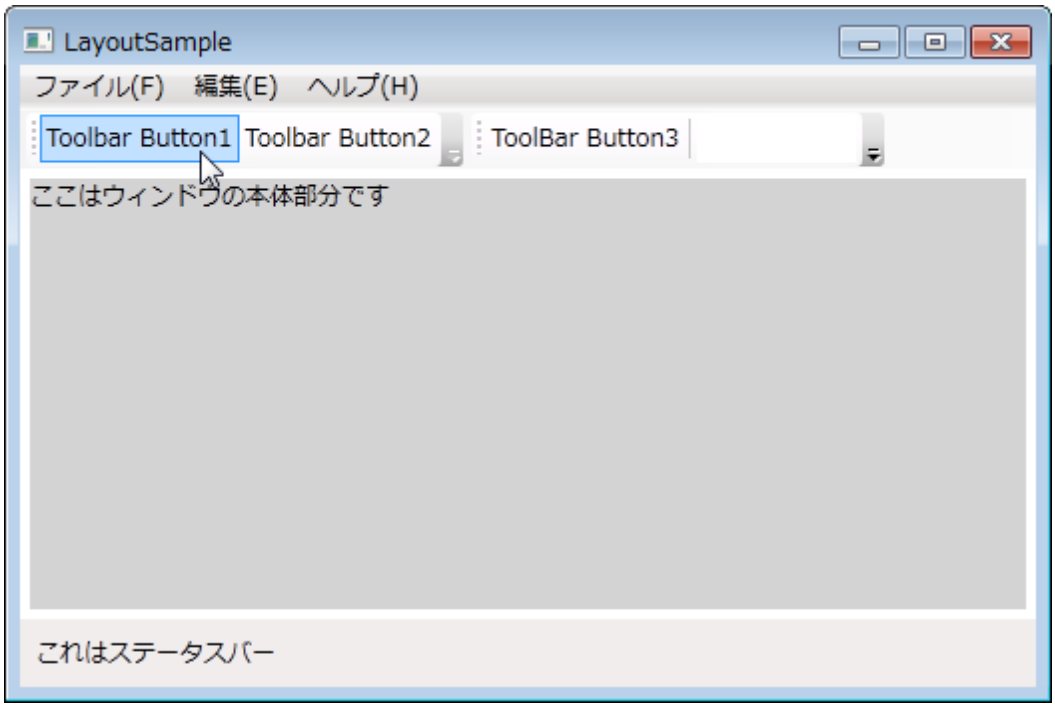
今回の総仕上げとして、これまでに紹介したレイアウトやコントロールを使って、メニューやツールバーを備えた一般的なウィンドウ（Windows エクスプローラーのようなウィンドウ）を作成してみましょう。具体的には、レイアウトとして Grid を使用し、内部にメニュー、ツールバー、ステータス バーを配置しています。

```
<Window x:Class="WpfApplication2.LayoutWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="LayoutSample" Height="350" Width="525">
  <Grid>
    <Grid.RowDefinitions>
      <RowDefinition Height="Auto"/>
      <!-- Menu -->
      <RowDefinition Height="Auto"/>
      <!-- ToolBar -->
      <RowDefinition Height="*"/>
      <!-- Body -->
      <RowDefinition Height="Auto"/>
      <!-- StatusBar -->
    </Grid.RowDefinitions>
    <!-- メニュー -->
    <Menu Grid.Row="0">
      <MenuItem Header="ファイル(_F)">
        <MenuItem Header="さらにメニュー">
          <MenuItem Header="サブメニュー項目#1"/>
          <MenuItem Header="サブメニュー項目#2"/>
        </MenuItem>
        <Separator/>
        <MenuItem Header="終了(_X)"/>
      </MenuItem>
      <MenuItem Header="編集(_E)">
        <MenuItem Header="コピー"/>
      </MenuItem>
      <MenuItem Header="ヘルプ(_H)">
        <MenuItem Header="○○○について" />
      </MenuItem>
    </Menu>

    <!-- ツールバー -->
    <ToolBarTray Grid.Row="1" >
      <ToolBar Name="toolBar1" Band="0" BandIndex="0">
        <Button>Toolbar Button1</Button>
        <Button>Toolbar Button2</Button>
      </ToolBar>
      <ToolBar Name="toolBar2" Band="0" BandIndex="1">
        <Button>ToolBar Button3</Button>
```

```
        <Separator/>
        <TextBox Name="textBox1" Width="80" />
        <Button>ToolBar Button4</Button>
      </ToolBar>
    </ToolBarTray>
    <WrapPanel Grid.Row="2" Margin="5,5,5,5" Background="LightGray">
      <!-- 本体部分 -->
      <TextBlock >ここはウィンドウの本体部分です</TextBlock>

    </WrapPanel>
    <!-- ステータスバー -->
    <StatusBar Grid.Row="3">
      <Label>これはステータスバー</Label>
    </StatusBar>
  </Grid>
</Window>
```

リスト 2-5 標準的な Windows アプリケーションのレイアウト例

実行すると次のようなウィンドウが表示されます。



もう 1 つ、TreeView コントロールを配置したウィンドウの例を紹介しましょう。基本的なレイアウト用のパネルとして DockPanel を使用します。TreeView の右側のペインにはエクスプローラーと同様、ListView を表示したいところですが、ListView については、「データ バインディング」の回で説明しま

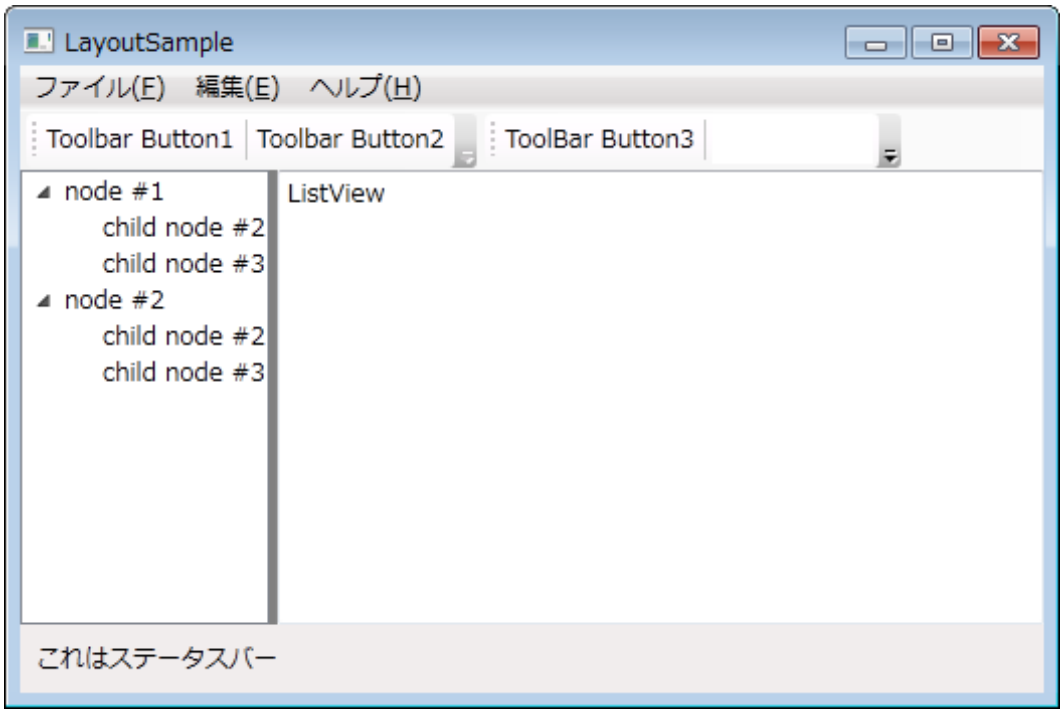
す。

```
<Window x:Class="WpfApplication2.NewLayoutWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="LayoutSample" Height="350" Width="525">
  <DockPanel>
    <!-- メニュー -->
    <Menu DockPanel.Dock="Top" >
      <MenuItem Header="ファイル(_F)">
        <MenuItem Header="さらにメニュー">
          <MenuItem Header="サブメニュー項目#1"/>
          <MenuItem Header="サブメニュー項目#2"/>
        </MenuItem>
        <Separator/>
        <MenuItem Header="終了(_X)"/>
      </MenuItem>
      <MenuItem Header="編集(_E)">
        <MenuItem Header="コピー"/>
      </MenuItem>
      <MenuItem Header="ヘルプ(_H)">
        <MenuItem Header="○○○について" />
      </MenuItem>
    </Menu>
    <!-- ツールバー -->
    <ToolBarTray DockPanel.Dock="Top">
      <ToolBar Name="toolBar1" Band="0" BandIndex="0">
        <Button>Toolbar Button1</Button>
        <Separator/>
        <Button>Toolbar Button2</Button>
      </ToolBar>
      <ToolBar Name="toolBar2" Band="0" BandIndex="1">
        <Button>ToolBar Button3</Button>
        <Separator/>
        <TextBox Name="textBox1" Width="80" />
        <Button>ToolBar Button4</Button>
      </ToolBar>
    </ToolBarTray>
    <!-- ステータスバー -->
    <StatusBar DockPanel.Dock="Bottom">
      <Label>これはステータスバー</Label>
```

```
    </StatusBar>
    <Grid>
      <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto"/>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*"/>
      </Grid.ColumnDefinitions>
      <TreeView Grid.Column="0" >
        <TreeViewItem Header="node #1" IsExpanded="True">
          <TreeViewItem Header="child node #2"/>
          <TreeViewItem Header="child node #3"/>
        </TreeViewItem>
        <TreeViewItem Header="node #2" IsExpanded="True">
          <TreeViewItem Header="child node #2"/>
          <TreeViewItem Header="child node #3"/>
        </TreeViewItem>
      </TreeView>
      <GridSplitter Grid.Column="1"
          HorizontalAlignment="Center"
          VerticalAlignment="Stretch"
          Background="Gray"
          ShowsPreview="True"
          Width="4"
        />
      <!-- ListView はデータバインディングの回で -->
      <TextBox Grid.Column="2">ListView</TextBox>
    </Grid>
  </DockPanel>
</Window>
```

リスト 2-6 Windows エクスプローラー風のウィンドウ

実行すると次のようなウィンドウが表示されます。



■まとめ

今回は、XAML で利用できるレイアウトと、代表的なコントロールについて学習しました。XAML では、パネルを用いることで、ウィンドウ上で絶対的な位置指定をしなくても、柔軟なレイアウトが実現できることがお分かりいただけたでしょう。

次回は、ボタンのクリックやメニュー選択といったコントロール操作に対応するロジックの記述（イベント処理）について紹介します。