

Imaging Library for Silverlight

2013.09.20 更新

グレースィティ株式会社

目次

製品の概要	2
ComponentOne Studio for Silverlight のヘルプ	2
The C1.Silverlight.Imaging.dll アセンブリ	2
Bitmap	3
主な特長	3
クイックスタート	3
手順 1: アプリケーションの作成	3
手順 2: 画像の追加	3-5
手順 3: コードの追加	5-6
手順 4: アプリケーションの実行	6
Bitmap の使い方	6-7
ツールバーの設定	7-8
ドラッグアンドドロップ動作の追加	8-9
画像の読み込みと保存	9
画像の印刷	9-10
ドラッグによるトリミング	10-13
画像のサイズ変更	13-14
元に戻す/やり直しの履歴	14-15
ワープ - 面白い機能	15
Image	16
主な特長	16
クイックスタート	16
手順 1: アプリケーションの作成	16-17
手順 2: 画像の追加	17
手順 3: アプリケーションの実行	17
XAML クイックリファレンス	17-18
タスク別ヘルプ	18
アニメーション画像を再生または停止する	18-20

製品の概要

Bitmap for Silverlight (C1Bitmap)を用いて、PNGやJPG形式の画像をロードして、ピクセル単位で編集したり、イメージタグで表示、ストリームへの保存を行うことができます。Image for Silverlight (C1Image)を使用して、従来のWebアプリケーションのように、Silverlight アプリケーションにアニメーションGIF形式の画像を表示できます。アニメーションGIFは、コンパクトであるため、アプリケーションにビジュアル要素を追加する簡易な動画として利用されています。

Image for Silverlight (C1Image) を使用して、従来のWebアプリケーションのように、Silverlight アプリケーションにアニメーションGIF形式の画像を表示できます。アニメーションGIFは、コンパクトであるため、アプリケーションにビジュアル要素を追加する簡易な動画として利用されています。

ComponentOne Studio for Silverlight のヘルプ

はじめに

ComponentOne Studio for Silverlight のすべてのコンポーネントで共通の使用方法については、「[ComponentOne Studio for Silverlight ユーザーガイド](#)」を参照してください。

The C1.Silverlight.Imaging.dll アセンブリ

C1.Silverlight.Imaging.dll アセンブリにはSilverlightのイメージング機能を強化するコントロールが含まれています。

主要なクラス

C1.Silverlight.Imaging.dll アセンブリには、次の主要なクラスが含まれます。

- **C1Image**: 標準の **System.Windows.Media.Image** クラスと同様で、アニメーションGIFにデザイン時サポートも提供します。標準の Image コントロールは、PNG 形式と JPEG 形式のみをサポートしています。
- **C1GifImage**: 標準の **System.Windows.Media.Imaging.BitmapImage** クラスと同様で、アニメーションGIFにデザイン時サポートも提供します。このクラスを使用するには、**Image** または **C1Image** 要素の **Source** プロパティに割り当てて安易に使用できます。
- **C1Bitmap**: プログラムによる画像の作成、PNG、JPG、およびGIFからのインポート/エクスポートする機能を提供します。

C1ImageMagnifier コントロールは以前C1.Silverlight.Imaging.dllアセンブリの一部でしたが、今はC1.Silverlight.Legacy.dllの一部です。このコントロールは、**Image**要素と同様ですが、追加で‘虫眼鏡’も提供しています。虫眼鏡の下の画像の一部がズームされます。ユーザーが簡単に画像の任意の部分上に虫眼鏡を移動してズームインすることができます。また、ズーム倍率を変更するためにキーボードを使用することができます。

Bitmap

Bitmap for Silverlight を使用すると、画像 (PNG および JPG) の読み込み、ピクセル単位の編集、イメージタグでの表示、ストリームへの保存を行うことができます。

主な特長

- **画像の縮小/トリミング**
ピクセルデータを編集することで、画像をサイズ変更したり、解像度を下げることができます。これにより、ファイルサイズが縮小し、アップロード時間が短縮します。また、Facebook などの Web ユーザーアカウントで画像をアップロードする際に、画像をトリミングして一部のみをアップロードできます。
- **プログラムによる画像編集**
C1Bitmap クラスを使用すると、個々のピクセルにアクセスして、特殊効果を適用したり、画像のトリミング、サイズ変更、任意の変換を行うことができます。
- **生成された画像を JPG/PNG として保存**
Silverlight では、**WritableBitmap** を使ってスクリーンショットを取得し、それを **C1Bitmap** に渡して、その結果を直ちに新しい PNG/JPG ファイルに保存できます。

クイックスタート

このクイックスタートは、**Bitmap for Silverlight** を初めて使用するユーザーのために用意されています。このクイックスタートでは、デフォルトの画像を読み込んでトリミングできる新しい Silverlight アプリケーションを作成します。この例では、Visual Studio 2010 と Silverlight 5 を使用します。

手順 1: アプリケーションの作成

この手順では、Visual Studio で、**Bitmap for Silverlight** を使用して、Silverlight アプリケーションを作成します。プロジェクトを設定し、**C1Bitmap** コントロールをアプリケーションに追加するには、次の手順に従います。

1. Visual Studio で、**[ファイル]→[新しいプロジェクト]** を選択します。
2. **[新しいプロジェクト]** ダイアログボックスで、左ペインから言語を選択し (この例では C# を使用)、右ペインのテンプレートリストから **[Silverlight アプリケーション]** を選択します。
3. プロジェクトの **[名前]** を入力し、**[OK]** をクリックします。**[新しい Silverlight アプリケーション]** ダイアログボックスが表示されます。
4. **[OK]** をクリックして、デフォルト設定を受け入れます。**MainPage.xaml** ファイルが開きます。
5. **[プロジェクト]→[参照の追加]** を選択します。**C1.Silverlight.5.dll** と **C1.Silverlight.Imaging.5.dll** を参照し、それらを選択したら、**[OK]** をクリックします。**ComponentOne Studio for Silverlight** を別の場所にインストールした場合は、その場所にインストールされます。

次の手順では、スタイルを設定し、プロジェクトに画像を追加します。

手順 2: 画像の追加

この手順では、画像のスタイルを設定し、新しい画像を作成するために、次の XAML を追加します。

1. 次の XAML を タグ内に追加し、デフォルトの タグを上書きします。

```
XAML
<UserControl.Resources>
  <Style x:Key="CE_SampleText" TargetType="TextBlock">
    <Setter Property="Foreground" Value="#FFF0F8FE" />
  </Style>
</UserControl.Resources>
```

```

    <Setter Property="FontWeight" Value="Normal" />
    <Setter Property="FontSize" Value="11" />
    <Setter Property="HorizontalAlignment" Value="Left"/>
    <Setter Property="VerticalAlignment" Value="Top"/>
    <Setter Property="Width" Value="400"/>
</Style>
<Style x:Key="CE_SampleTextBkg" TargetType="Border">
    <Setter Property="Background">
        <Setter.Value>
            <LinearGradientBrush EndPoint="1,0.5" StartPoint="0,0.5">
                <GradientStop Color="#99071D2E" Offset="0.003"/>
                <GradientStop Color="#00071D2E" Offset="1"/>
            </LinearGradientBrush>
        </Setter.Value>
    </Setter>
    <Setter Property="CornerRadius" Value="2"/>
    <Setter Property="Padding" Value="5 0 0 0"/>
</Style>
<SolidColorBrush Color="#55FFFFFF" x:Key="MaskBrush"/>
</UserControl.Resources>
<Grid x:Name="LayoutRoot" Background="White">
    <Border BorderBrush="#FF8FB4CC" BorderThickness="3" Grid.Row="2"
VerticalAlignment="Center" HorizontalAlignment="Center">
        <Grid Name="imageGrid">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="*" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <Image Stretch="None" Name="image" Grid.RowSpan="3"
Grid.ColumnSpan="3"/>
            <Grid Name="topMask" Grid.ColumnSpan="2" Background="{StaticResource
MaskBrush}" />
            <Grid Name="bottomMask" Grid.Column="1" Grid.Row="2"
Grid.ColumnSpan="2" Background="{StaticResource MaskBrush}" />
            <Grid Name="leftMask" Grid.RowSpan="2" Grid.Row="1" Background="
{StaticResource MaskBrush}" />
            <Grid Name="rightMask" Grid.Column="2" Grid.RowSpan="2" Background="
{StaticResource MaskBrush}" />
        </Grid>
    </Border>
</Grid>

```

2. プロジェクトに画像を追加します。

- [プロジェクト]→[既存のアイテムの追加]を選択します。
- 画像を参照します。この例では、**ComponentOne Studio for Silverlight** に付属する C1.Silverlight.Imaging\Crop サンプルの画像 Lenna.jpg を使用します。
- この画像を選択し、[追加]をクリックします。

次の手順では、画像のトリミングに使用するコードを追加します。

手順 3: コードの追加

この手順のコードは、デフォルトの画像を読み込み、ユーザーがその画像をトリミングできるようにします。次の手順に従います。

1. MainPage.xaml.cs ファイルを開き、次の **using** (Visual Basic の場合は **Imports**) 文を追加します。

```
C#  
  
using Cl.Silverlight;  
using Cl.Silverlight.Imaging;  
using System.IO;
```

2. デフォルトの画像を読み込み、トリミングを定義するために、次のコードを追加します。

```
C#  
  
public partial class MainPage : UserControl  
{  
    ClBitmap bitmap = new ClBitmap();  
    Rect selection;  
    public MainPage()  
    {  
        InitializeComponent();  
        LoadDefaultImage();  
        image.Source = bitmap.ImageSource;  
        var mouseHelper = new ClMouseHelper(imageGrid);  
        mouseHelper.MouseDragStart += OnDrag;  
        mouseHelper.MouseDragMove += OnDrag;  
    }  
    void OnDrag(object sender, MouseDragEventArgs e)  
    {  
        var transform = Application.Current.RootVisual.TransformToVisual(image);  
        var start = transform.Transform(e.StartPosition);  
        var end = transform.Transform(e.CurrentPosition);  
        start.X = Math.Min(Math.Max(start.X, 0), bitmap.Width);  
        end.X = Math.Min(Math.Max(end.X, 0), bitmap.Width);  
        start.Y = Math.Min(Math.Max(start.Y, 0), bitmap.Height);  
        end.Y = Math.Min(Math.Max(end.Y, 0), bitmap.Height);  
        selection = new Rect(new Point(  
            Math.Round(Math.Min(start.X, end.X)),  
            Math.Round(Math.Min(start.Y, end.Y))),  
            new Size(Math.Round(Math.Abs(start.X - end.X)),  
                Math.Round(Math.Abs(start.Y - end.Y))));  
        UpdateMask();  
    }  
    void UpdateMask()  
    {  
        topMask.Height = selection.Top;  
        bottomMask.Height = bitmap.Height - selection.Bottom;  
        leftMask.Width = selection.Left;  
        rightMask.Width = bitmap.Width - selection.Right;  
    }  
    void LoadDefaultImage()
```

```

    {
        LoadImageStream(Application.GetResourceStream(new
Uri("/SilverlightApplication1;component/Lenna.jpg", UriKind.Relative)).Stream);
    }
    void LoadImageStream(Stream stream)
    {
        bitmap.SetStream(stream);
        imageGrid.Width = bitmap.Width;
        imageGrid.Height = bitmap.Height;
        selection = new Rect(0, 0, bitmap.Width, bitmap.Height);
        UpdateMask();
    }
}

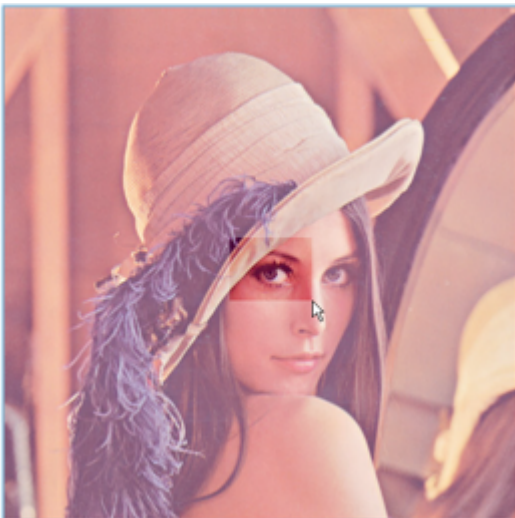
```

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

1. [デバッグ]メニューから[デバッグ開始]を選択して、画像を表示します。
2. 画像をクリックし、マウスの左ボタンを押しながらポインタをドラッグします。**ComponentOne Studio for Silverlight** に付属する Crop サンプルは、トリミングされた画像をエクスポートし、その画像をファイルに保存する方法を示します。

次の図では、目の領域がトリミングされています。



おめでとうございます!

これで、Image for Silverlight クイックスタートは終了です。

Bitmap の使い方

Silverlight 5 が正式にリリースされ、さまざまな機能をすぐに使用できるようになりました。たとえば、ドラッグアンドドロップ管理、マウスの右ボタンクリックの処理、印刷などがあります。これらの機能のほとんどは、ComponentOne Studio の Silverlight 3 用コントロールを使って実行できましたが、Silverlight 5 では、フレームワーク自体の限界を広げることで、Studio の価値や柔軟性をさらに高めています。

このサンプルでは、新しい Silverlight 5 の機能と従来の ComponentOne Silverlight コントロールをいくつか組み合わせて、一般的なアプリケーションである画像エディタを拡張します。**C1Bitmap** は、クライアント側で画像を処理できる完全に管理可能なビットマップ API です。このサンプルでは、**C1Bitmap** のさまざまな機能と Silverlight 5 の新しい機能を組み合わせて、完全な画像編集アプリケーションを構築します。

Imaging Library for Silverlight

使用する Silverlight 5 の機能

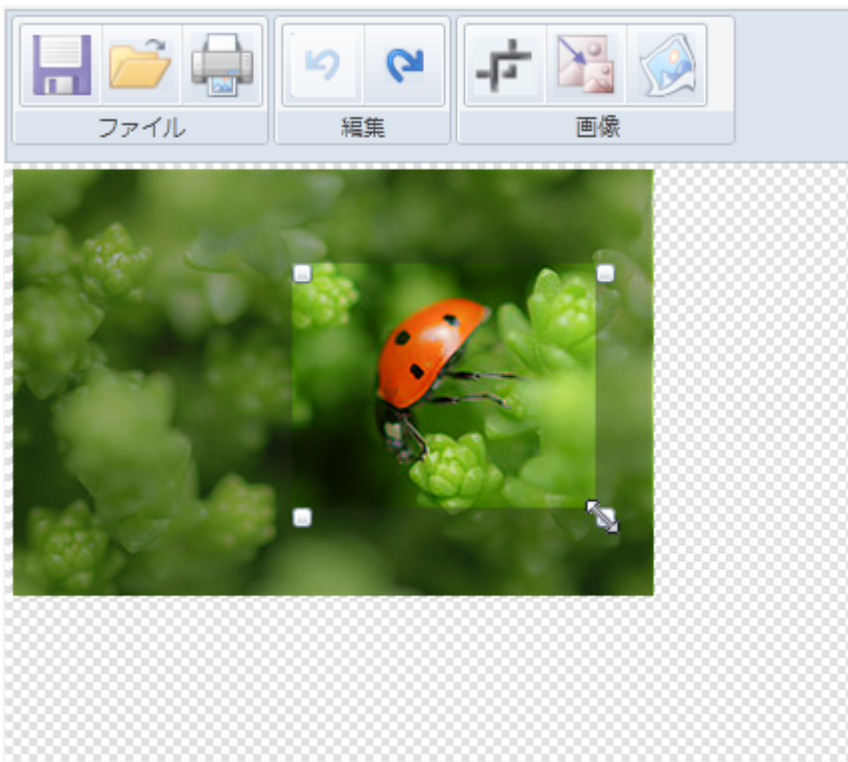
- 外部ソースから Silverlight へのドラッグアンドドロップ
- 印刷

ユーザーは、ドキュメントフォルダなどの外部ソースから画像をドラッグし、Silverlight 画像エディタにドロップできるようになります。エディタにドロップした画像は、印刷できるようになります。

C1Bitmap を使って実装する機能

- トリミング
- サイズ変更
- ワープ
- 元に戻す/やり直し

このアプリケーションを作成するには、C1Toolbar も使用します。



ツールバーの設定

ここで作成するアプリケーションは、上部に C1Toolbar があり、残りの部分は格子模様の背景になります。C1Toolbar と C1ToolbarStrip を使用して、リボン形式のグループを含むツールバーを作成します。**ComponentOne Studio for Silverlight** には、C1CheckeredBorder という便利な格子模様のパネルコントロールがあり、C1.Silverlight.Extended ライブラリに置かれています。このコントロールは、画像エディタのデザインに役立ちます。ツールバーは、3つの **C1ToolbarGroups** (File, Edit, Image) で構成されます。

XAML

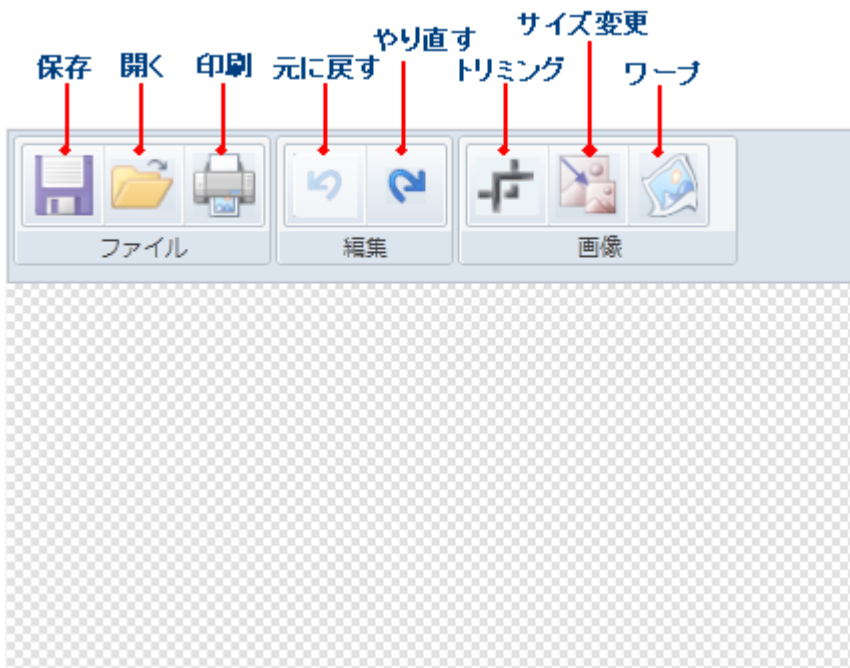
```
<c1:C1Toolbar Name="c1Toolbar1" Grid.Row="0">
  <c1:C1ToolbarGroup Header="ファイル">
    <c1:C1ToolbarStrip >
      <c1:C1ToolbarButton Name="btnSave" Click="btnSave_Click">
        <Image Source="Resources/save.png" Margin="2"
ToolTipService.ToolTip="保存"/>
```



```

        </cl:C1ToolBarButton>
        <cl:C1ToolBarButton Name="btnOpen" Click="btnOpen_Click">
            <Image Source="Resources/Open.png" Margin="2"
ToolTipService.ToolTip="開く"/>
        </cl:C1ToolBarButton>
        <cl:C1ToolBarButton Name="btnPrint" Click="btnPrint_Click">
            <Image Source="Resources/Print.png" Margin="2"
ToolTipService.ToolTip="印刷"/>
        </cl:C1ToolBarButton>
    </cl:C1ToolBarStrip>
</cl:C1ToolBarGroup>
...
</cl:C1ToolBar>

```



ドラッグアンドドロップ動作の追加

Silverlight 5 では、すべてのコントロールに、固有のドラッグアンドドロップイベント (DragOver、DragLeave、Drop など) が用意されています。これを使用して、事実上あらゆる要素間のドラッグアンドドロップ処理を管理できます。さらに重要なこととして、Silverlight 5 では、Silverlight アプリケーションの外部からも項目をドラッグできます。必要な作業は、コンテナの **AllowDrop** プロパティを **True** に設定したら、**Drop** イベントを処理してファイルを取得し、目的の作業を行うだけです。このサンプルでは、[Microsoft 提供](#)のサンプルのコードを使用します。ドロップ可能なコントロールとして、C1CheckedBorder コントロールを使用します。

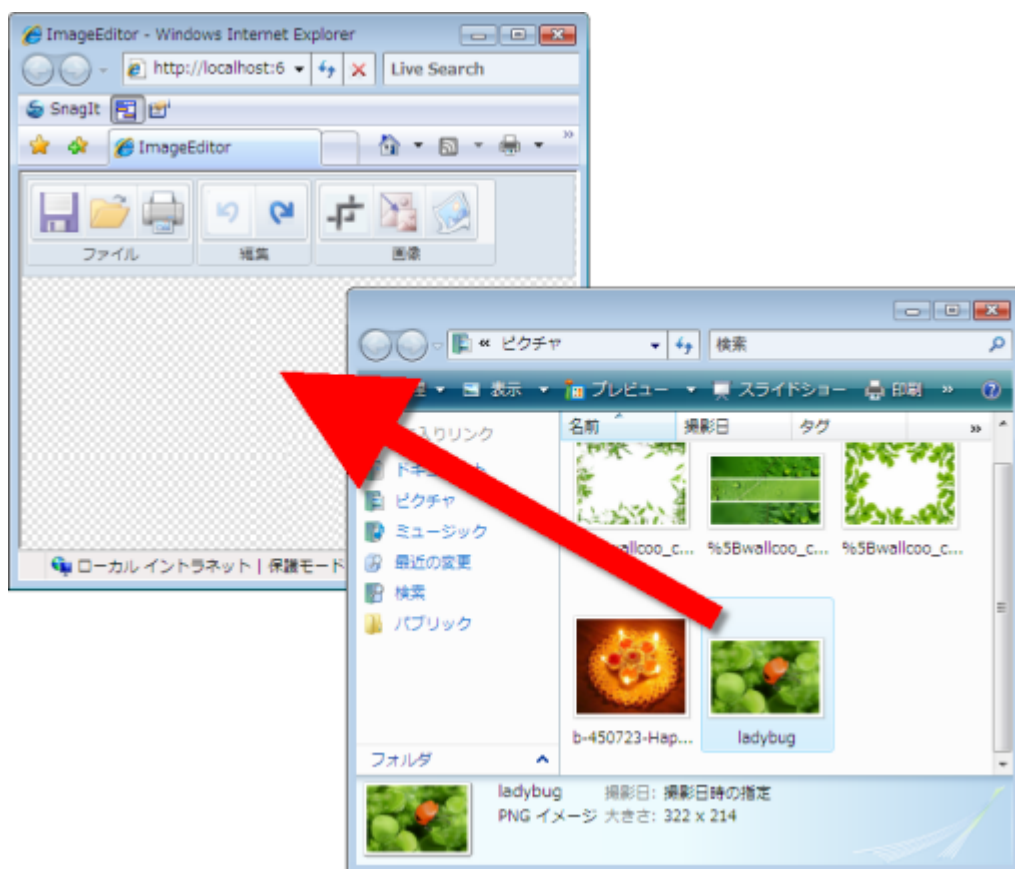
```

C#
<cl:C1CheckedBorder Name="checkeredBack" AllowDrop="True" Drop="DropTarget_Drop" />
private void DropTarget_Drop(object sender, DragEventArgs e)
{
    // DragEventArgs から FileInfo 配列を取得します
    IDataObject dataObject = e.Data;
    var files = (FileInfo[])dataObject.GetData(DataFormats.FileDrop);
    //最初にファイルを取得します
    if (files != null)

```

```
{  
    FileInfo file = files[0];  
    if (IsImageFile(file.Extension))  
    {  
        Stream stream = file.OpenRead();  
        LoadImageStream(stream);  
    }  
}
```

コンピュータから画像ファイルを Web ブラウザにドラッグすると、ポインタがドロップを表すポインタに変わります。このサンプルは、(画像の拡張子をチェックした上で)1つのファイルのみを受け入れますが、Microsoft のサンプルのように、一度に複数のファイルをドロップすることも簡単にできます。



画像の読み込みと保存

このアプリケーションの2つの重要な操作は、開くと保存です。ここでは、ユーザーがこのアプリケーションに画像ファイルをドロップする場合と、コンピュータを参照してファイルを開く場合の両方で機能するように画像を開くメソッドを設定します。それには、入力として単純なストリームを渡し、ストリームから画像を読み込むようにアプリケーションを設定します。また、組み込みの **OpenFileDialog** および **SaveFileDialogs** を使用して、アプリケーションからユーザーのコンピュータ内にあるファイルにアクセスできるようにします。読み込んだ画像は、ページ内の標準 Image コントロールに表示すると共に、バックグラウンドで **C1Bitmap** コンポーネントにも読み込みます。これで、トリミング、サイズ変更、ワープなどのアクションを使用して、画像を操作できるようになります。

画像の印刷

画像が読み込まれたら、簡単に印刷を実行できます。Silverlight 5 の印刷機能は、ほとんどすべての Silverlight アプリケー

ションで利用できます。基本的に、印刷機能には **PrintDocument** コンポーネントが含まれます。ここでは、印刷用のコードも、ドラッグアンドドロップで使用した Microsoft 提供のサンプルに倣ってモデル化します。印刷するには、**PrintDocument.Print** メソッドを呼び出し、**PrintPage** イベントで、ドキュメントの **PageVisual** プロパティを任意の UI 要素に設定します。ここでは、画像コンテナに設定しています。画像と一緒にツールバー全体を印刷することもできますが、ここでは無意味です。

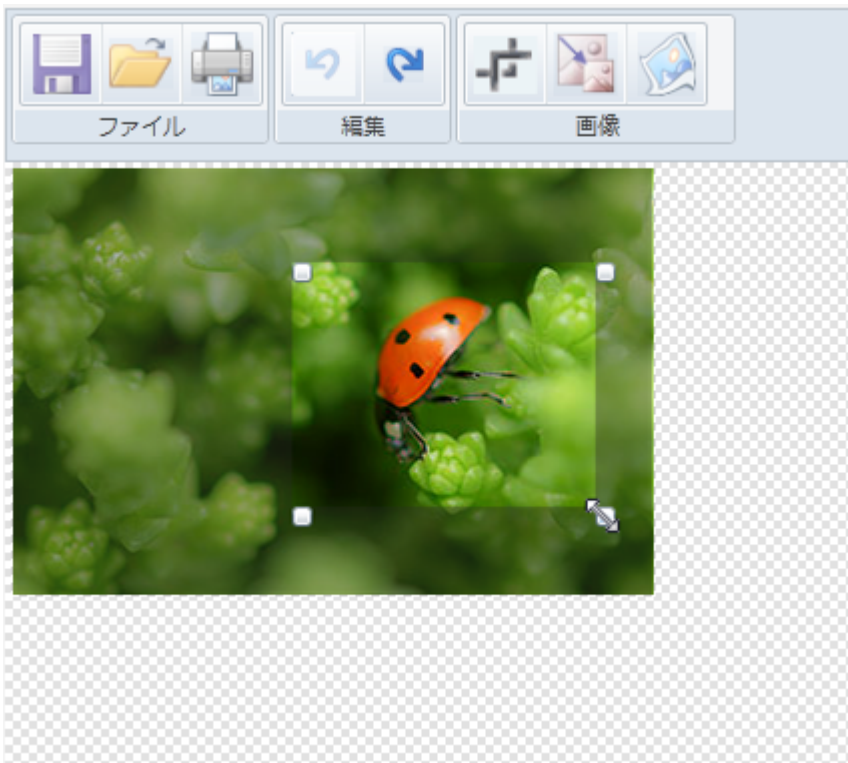
C#

```
PrintDocument printDocument = new PrintDocument();
private void btnPrint_Click(object sender, RoutedEventArgs e)
{
    printDocument.Print("My Image");
}
void printDocument_PrintPage(object sender, PrintPageEventArgs e)
{
    e.PageVisual = imageGrid;
    e.HasMorePages = false;
}
```

ドラッグによるトリミング

クライアント側で画像を完全にトリミングできれば、とても便利なタスクになります。Silverlight では、**C1Bitmap** または **WritableBitmap** クラス (Silverlight 3 で導入) を使用してこれを実現できます。**C1Bitmap** コンポーネントには、あらゆるビットマップ関連操作を実行する際に簡単に使用できる API が用意されています。この API は、単純に色を取得および設定することができ、**GetPixel** および **SetPixel** メソッドを使ってピクセルに直接アクセスできます。

C1Bitmap は、画像をトリミングするために必要な API を提供しますが、UI は提供しません。画像トリミングの UI を実装する方法は、数え切れないほどあります。ほとんどの開発者や画像編集者は、装飾付きのボックスをドラッグする方法がよいと考えるでしょう。これは、Adobe Photoshop などの市販の画像編集ソフトウェアでよく使用されている方法です。ここでも、このようなボックスを作成します。



次の XAML は、トリミングボックスを作成するために必要な要素を定義します。このボックスは、ユーザーがドラッグできる4つ

Imaging Library for Silverlight

のつまみ (Thumb) と、トリミング領域をマスクする4つの影付き四角形 (Rectangle) で構成されます。これらの要素の位置をコードで簡単に調整できるように Canvas 内に配置します。

XAML

```
<Canvas Name="cropCanvas">
    <Rectangle Name="topMask" Fill="{StaticResource MaskBrush}" Canvas.Top="0"
Canvas.Left="0" />
    <Rectangle Name="bottomMask" Fill="{StaticResource MaskBrush}" />
    <Rectangle Name="leftMask" Fill="{StaticResource MaskBrush}" Canvas.Left="0"/>
    <Rectangle Name="rightMask" Fill="{StaticResource MaskBrush}" Canvas.Top="0" />
    <Thumb Name="cropUL" Width="10" Height="10" DragDelta="cropUL_DragDelta"
Cursor="SizeNWSE" />
    <Thumb Name="cropUR" Width="10" Height="10" DragDelta="cropUR_DragDelta"
Cursor="SizeNESW" />
    <Thumb Name="cropBL" Width="10" Height="10" DragDelta="cropBL_DragDelta"
Cursor="SizeNESW" />
    <Thumb Name="cropBR" Width="10" Height="10" DragDelta="cropBR_DragDelta"
Cursor="SizeNWSE" />
</Canvas>
```



トリミングボックスの操作に必要なコードは、かなり複雑です。ドラッグ可能なトリミングボックスは、動作と Visual State Manager を使って実装することもできますが、Silverlight の初級開発者にとっては、コードを使用した方が間違いなく理解しやすいソリューションになります。トリミングボックス UI の目的は、**C1Bitmap** がトリミング座標を決定するために使用する単純な四角形 (Rect) を作成することです。ツールバーの [Crop] ボタンをクリックすると、トリミングボックスがフルサイズで表示されます。ユーザーがつまみをドラッグした場合は、各 Thumb の **DragDelta** イベントを使用して、垂直および水平方向の移動をキャプチャします。次に、少しのロジックと単純な計算を使用して、ユーザーがドラッグしていないつまみの動作を操作します。トリミングアクションを完了するには、**[Crop]** ボタン (**C1ToolBarToggleButton**) を再度クリックします。

C#

```
private void cropUL_DragDelta(object sender,
System.Windows.Controls.Primitives.DragDeltaEventArgs e)
{
    double left = Canvas.GetLeft(cropUL) + e.HorizontalChange;
    double top = Canvas.GetTop(cropUL) + e.VerticalChange;
    if (left > 0 && left < bitmap.Width && cropBox.Width > e.HorizontalChange)
    {
        cropBox = new Rect(left, cropBox.Top, cropBox.Width - e.HorizontalChange,
cropBox.Height);
    }
    if (top > 0 && top < bitmap.Height && cropBox.Height > e.VerticalChange)
    {
        cropBox = new Rect(cropBox.Left, top, cropBox.Width, cropBox.Height -
e.VerticalChange);
    }
    UpdateCropBox();
}
private void cropUR_DragDelta(object sender,
System.Windows.Controls.Primitives.DragDeltaEventArgs e)
{
    double left = Canvas.GetLeft(cropUR) + e.HorizontalChange;
    double top = Canvas.GetTop(cropUR) + e.VerticalChange;
```

```

    if (left > 0 && left < bitmap.Width && left > cropBox.Left)
    {
        cropBox = new Rect(cropBox.Left, cropBox.Top, left - cropBox.Left,
cropBox.Height);
    }
    if (top > 0 && top < bitmap.Height && cropBox.Height > e.VerticalChange)
    {
        cropBox = new Rect(cropBox.Left, top, cropBox.Width, cropBox.Height -
e.VerticalChange);
    }
    UpdateCropBox();
}
private void cropBL_DragDelta(object sender,
System.Windows.Controls.Primitives.DragDeltaEventArgs e)
{
    double left = Canvas.GetLeft(cropBL) + e.HorizontalChange;
    double top = Canvas.GetTop(cropBL) + e.VerticalChange;
    if (left > 0 && left < bitmap.Width && cropBox.Width > e.HorizontalChange)
    {
        cropBox = new Rect(left, cropBox.Top, cropBox.Width - e.HorizontalChange,
cropBox.Height);
    }
    if (top > 0 && top < bitmap.Height && top > cropBox.Top)
    {
        cropBox = new Rect(cropBox.Left, cropBox.Top, cropBox.Width, top -
cropBox.Top);
    }
    UpdateCropBox();
}
private void cropBR_DragDelta(object sender,
System.Windows.Controls.Primitives.DragDeltaEventArgs e)
{
    double left = Canvas.GetLeft(cropBR) + e.HorizontalChange;
    double top = Canvas.GetTop(cropBR) + e.VerticalChange;

    if (left > 0 && left < bitmap.Width && left > cropBox.Left)
    {
        cropBox = new Rect(cropBox.Left, cropBox.Top, left - cropBox.Left,
cropBox.Height);
    }
    if (top > 0 && top < bitmap.Height && cropBox.Height + e.VerticalChange > 0)
    {
        cropBox = new Rect(cropBox.Left, cropBox.Top, cropBox.Width, cropBox.Height +
e.VerticalChange);
    }
    UpdateCropBox();
}

```

上の "if" 文で、つまみの範囲に対していくつかのロジックを適用します。たとえば、右下のつまみを左下のつまみより左にドラッグできないようにします。また、どのつまみも画像の外までドラッグできないようにします。

C#

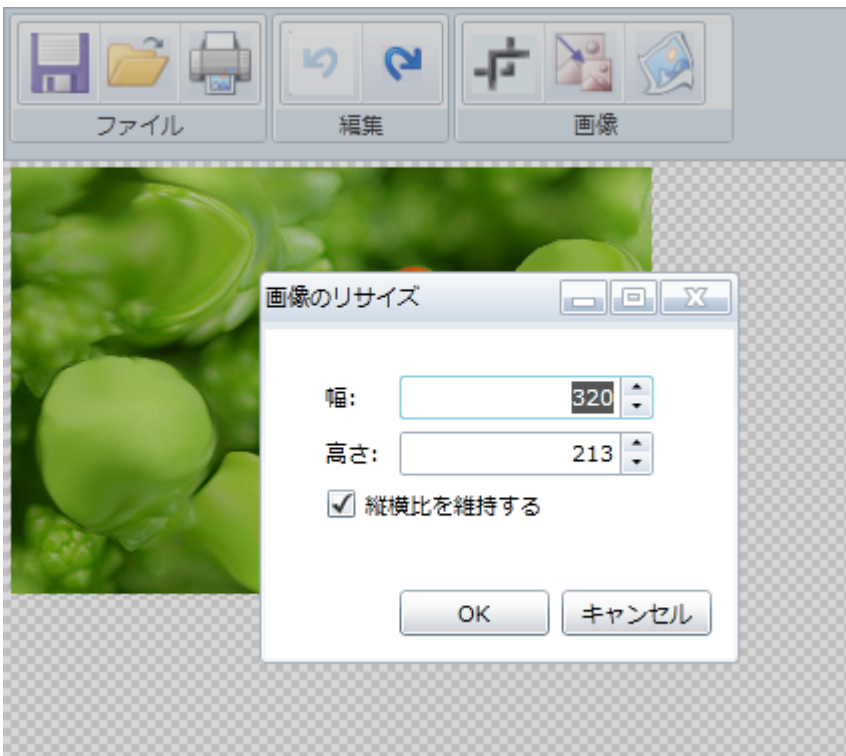
```
private void UpdateCropBox ()
{
    Canvas.SetLeft (cropUL, cropBox.Left);
    Canvas.SetTop (cropUL, cropBox.Top);
    Canvas.SetLeft (cropUR, cropBox.Left + cropBox.Width);
    Canvas.SetTop (cropUR, cropBox.Top);
    Canvas.SetLeft (cropBL, cropBox.Left);
    Canvas.SetTop (cropBL, cropBox.Top + cropBox.Height);
    Canvas.SetLeft (cropBR, cropBox.Left + cropBox.Width);
    Canvas.SetTop (cropBR, cropBox.Top + cropBox.Height);
    UpdateMask ();
    cropping = true;
}
```

UpdateCropBox メソッドは、**cropBox** Rect の **Left**、**Top**、**Width**、および **Height** プロパティに基づいて、すべての **cropCanvas** 要素の位置を更新します。最終的にトリミングを適用 (**[Crop]** ボタンを再クリック) する段階になると、**C1Bitmap** を使用します。ここで、境界 Rect 内のピクセルを取得して新しい **C1Bitmap** にコピーし、元の画像を新しい画像に置き換えます。

```
C#
private void CropImage ()
{
    bitmap2 = new C1Bitmap ((int) cropBox.Width, (int) cropBox.Height);
    bitmap2.BeginUpdate ();
    for (int x = 0; x < cropBox.Width; ++x)
    {
        for (int y = 0; y < cropBox.Height; ++y)
        {
            bitmap2.SetPixel (x, y, bitmap.GetPixel (x + (int) cropBox.X, y +
(int) cropBox.Y));
        }
    }
    bitmap2.EndUpdate ();
    bitmap.Copy (bitmap2, false);
    UpdateImage (true);
    InitCropHandles ();
}
```

画像のサイズ変更

2つ目の便利な画像編集タスクは、画像のサイズ変更(拡大縮小)です。C1Bitmap は、画像を簡単にサイズ変更する方法を備えています。ここで行う作業は、ユーザーからの入力をキャプチャするための UI を作成するだけです。このサンプルでは、画像の新しい Width と Height を入力するように求める子ウィンドウを表示します。次に、これらの値をサイズ変更メソッドに渡すと、既存の画像が新しいサイズの画像に置き換えられます。このサイズ変更アクションは、縦横比を固定できるとよいですが、自由にサイズ変更できるようにもします。これはすべて UI を介して処理できます。Bitmap コンポーネントには依存しません。



C#

```
void ResizeImage(int w, int h)
{
    bitmap = new C1Bitmap(bitmap, w, h);
    UpdateImage(true);
}
```

元に戻す/やり直しの履歴

誰でもミスはします。どのようなアプリケーションでも、間違いを元に戻す(さらにやり直す)ことができれば、最初からやり直す手間が省けるのでたいへん便利です。このサンプルでは、画像が変更されるたびに画像のコピーを3つまで保存するという簡単な方法で、**Undo/Redo** を実現します。変更には、トリミング、サイズ変更、ワープがあります。ここでは、変更履歴を前後に移動すると共に、ユーザーがさらに変更を追加できるようにするテクニックを示します。実際のコードは、**C1Bitmap** または Silverlight 5 の拡張機能とは無関係なため、任意のコントロールを元に戻す/やり直すために使用できます。

C#

```
List<C1Bitmap> undoBitmaps = new List<C1Bitmap>();
List<C1Bitmap> redoBitmaps = new List<C1Bitmap>();
private void btnUndo_Click(object sender, RoutedEventArgs e)
{
    if (undoBitmaps.Count > 1)
    {
        bitmap = new C1Bitmap(undoBitmaps.ElementAt(undoBitmaps.Count - 2));
        redoBitmaps.Add(new C1Bitmap(undoBitmaps.ElementAt(undoBitmaps.Count - 1)));
        undoBitmaps.RemoveAt(undoBitmaps.Count - 1);
        UpdateImage(false);
    }
    UpdateEditButtons();
}
```

Imaging Library for Silverlight

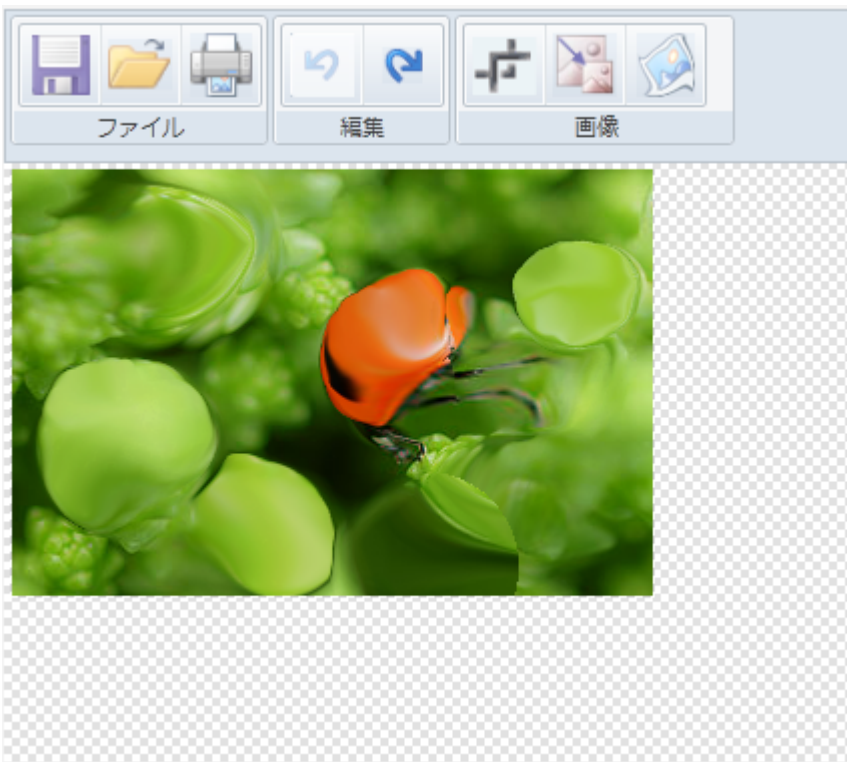
```
private void btnRedo_Click(object sender, RoutedEventArgs e)
{
    if (redoBitmaps.Count > 0)
    {
        bitmap = new C1Bitmap(redoBitmaps.ElementAt(redoBitmaps.Count - 1));
        undoBitmaps.Add(new C1Bitmap(redoBitmaps.ElementAt(redoBitmaps.Count - 1)));
        redoBitmaps.RemoveAt(redoBitmaps.Count - 1);
        UpdateImage(false);
    }
    UpdateEditButtons();
}

void UpdateHistory()
{
    //現在のビットマップをメモリに追加します
    undoBitmaps.Add(new C1Bitmap(bitmap));
    redoBitmaps.Clear();

    //元に戻す/やり直しの履歴として C1Bitmap に加えられた変更を3つまで保持できるように制限します
    if (undoBitmaps.Count > 4)
        undoBitmaps.RemoveAt(0);
    UpdateEditButtons();
}
```

ワープ - 面白い機能

C1Bitmap コンポーネントの当初のデモでは、ワープを使ってピクセル単位で画像を操作する方法を示しました。これは、現在 Control Explorer で見ることができます。このコードは、基本的に多くの計算を使用して、画像ピクセルに円形の変換を適用します。このサンプルではこのコードに変更を加えていません。面白いので試してみてください。



Image

Image for Silverlight を使用すると、従来の Web アプリケーションと同様に、Silverlight ページでアニメーション GIF 画像を表示できます。アニメーション GIF はコンパクトな上、アプリケーションに最小限の作業で魅力的なビジュアル要素を追加できます。

主な特長

以下に、**Image for Silverlight** の便利な機能をいくつか示します。

- アニメーション GIF ファイルのサポート**
 Image を使用すると、Silverlight アプリケーションにアニメーション GIF ファイルを追加できます。**C1Image** コントロールを使用して、設計時に GIF 画像を追加できます(標準のイメージコントロールは、PNG 形式と JPEG 形式のみをサポート)。
- 再生、一時停止、停止のメソッド**
C1Image コントロールで使用されるイメージソースは**C1GifImage** クラスです。このクラスは、メディアプレイヤーと同様のコマンドを提供して、GIF アニメーションをプログラムで制御できるようにします。これらのメソッドを使用して、タスクを実行しながら GIF をアニメーション表示することで、面白い進捗状況インジケータを作成したり、アプリケーションの状態に合わせたアニメーションを作成することができます。
- 他の使用例を見る**
 Photobucket は、Image コントロールを使用して、Silverlight サイトで GIF 画像をロードしています。[ここをクリックしてください](#)。

クイックスタート

このクイックスタートは、**Image for Silverlight** を初めて使用するユーザーのために用意されています。このクイックスタートでは、Visual Studio で新しいプロジェクトを作成し、**C1Image** コントロールをアプリケーションに追加してから、アプリケーションを実行します。この例では、Visual Studio 2010 と Silverlight 5 を使用します。

手順 1: アプリケーションの作成

この手順では、Visual Studio で、**Image for Silverlight** を使用して、Silverlight アプリケーションを作成します。

プロジェクトを設定し、**C1Image** コントロールをアプリケーションに追加するには、次の手順に従います。

- Visual Studio で、**[ファイル]→[新しいプロジェクト]**を選択します。
- [新しいプロジェクト]**ダイアログボックスで、左ペインから言語を選択し(この例では C# を使用)、右ペインのテンプレートリストから**[Silverlight アプリケーション]**を選択します。
- プロジェクトの**[名前]**を入力し、**[OK]**をクリックします。**[新しい Silverlight アプリケーション]**ダイアログボックスが表示されます。
- 必要に応じて、**[Silverlight アプリケーションを新しい Web サイトでホストする]**ボックスをオフにし、**[OK]**をクリックします。**MainPage.xaml** ファイルが開きます。
- ツールボックスで、**[C1Image]**アイコンをダブルクリックして、**C1Image** コントロールを **MainPage.xaml** に追加します。XAML マークアップは次のようになります。

XAML

```
<UserControl x:Class="C1Image.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d">
```

Imaging Library for Silverlight

```
d:DesignHeight="300" d:DesignWidth="400"
xmlns:my="clr-namespace:C1.Silverlight.Imaging;assembly=C1.Silverlight.Imaging">

    <Grid x:Name="LayoutRoot" Background="White">
        <my:C1Image HorizontalAlignment="Left" Margin="170,81,0,0"
Name="c1Image1" VerticalAlignment="Top" />
    </Grid>
</UserControl>
```

C1.Silverlight.Imaging 名前空間とタグがプロジェクトに追加されていることがわかります。

次の手順では、コントロールに画像を追加します。

手順 2: 画像の追加

次に、**C1Image** コントロールに画像を追加します

1. **C1Image** コントロールを選択し、Visual Studio の **[プロパティ]** ウィンドウで、**Source** プロパティの横にある省略符ボタンをクリックします。**[画像の選択]** ダイアログボックスが開きます。
2. [追加] ボタンをクリックします。
3. **[開く]** ダイアログボックスで、画像を参照して見つけます。.gif (アニメーションまたは静止画像)、.jpg、.jpeg、または .png を選択できます。
4. 画像を選択し、**[開く]** をクリックします。
5. **[OK]** をクリックします。

次の手順では、このアプリケーションを実行します。

手順 3: アプリケーションの実行

C1Image コントロールを含む Silverlight アプリケーションを作成したので、このアプリケーションを実行します。

[デバッグ] メニューから **[デバッグ開始]** を選択して、画像を表示します。



おめでとうございます。これで、**Image for Silverlight** クイックスタートは終了です。

XAML クイックリファレンス

このトピックでは、**C1Image** コントロールの作成に使用される XAML の概要を提供します。

開発を開始するには、ルート要素タグに **c1** 名前空間宣言を追加します。

XAML

```
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
```

次は、**C1Imaging_Demo** サンプルにある **C1Image** のサンプルです。



このサンプルの XAML は次のようになります。

XAML

```
<UserControl x:Class="C1Imaging_Demo.DemoGifImage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:climaging="http://schemas.componentone.com/winfx/2006/xaml"
  Background="#FF374F5D">
  <Grid x:Name="LayoutRoot" Background="Transparent">
    <Path Height="200" Width="220"
      Stretch="Fill"
      Data="M 61.3431396484375,0 C61.3431396484375,0
80.2991943359375,38.40919494628906 80.2991943359375,38.40919494628906
80.2991943359375,38.40919494628906 122.686279296875,44.56840515136719
122.686279296875,44.56840515136719 122.686279296875,44.56840515136719
92.01470947265625,74.46580505371094 92.01470947265625,74.46580505371094
92.01470947265625,74.46580505371094 99.25527954101562,116.68159484863281
99.25527954101562,116.68159484863281 99.25527954101562,116.68159484863281
61.3431396484375,96.75 61.3431396484375,96.75 61.3431396484375,96.75
23.430999755859375,116.68159484863281 23.430999755859375,116.68159484863281
23.430999755859375,116.68159484863281 30.67156982421875,74.46580505371094
30.67156982421875,74.46580505371094 30.67156982421875,74.46580505371094
0,44.56840515136719 0,44.56840515136719 0,44.56840515136719
42.3870849609375,38.40919494628906 42.3870849609375,38.40919494628906
42.3870849609375,38.40919494628906 61.3431396484375,0 61.3431396484375,0 z"
      Fill="Black" Stroke="#FF8FB4CC" StrokeThickness="2.5" />
    <climaging:C1Image x:Name="image" VerticalAlignment="Center"
  HorizontalAlignment="Center" Stretch="None" />
  </Grid>
</UserControl>
```

タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio でのプログラミングに精通しており、**C1Image** コントロールの一般的な使用方法を理解していることを前提としています。**Image for Silverlight** 製品を初めて使用される場合は、まず「[クイックスタート](#)」を参照してください。

このセクションの各トピックは、**Image for Silverlight** 製品を使って特定のタスクを行うための方法を提供します。また、各トピックは、新しい Silverlight プロジェクトが既に作成されていることを前提としています。

アニメーション画像を再生または停止する

C1Image コントロールで使用されるイメージソースは **C1GifImage** クラスです。このクラスは、メディアプレイヤーと同様のコマンドを提供します。**Play**、**Stop**、および **Pause** メソッドを使用して、GIF アニメーションをプログラムで制御できます。**Play** メソッドと **Stop** メソッドの使用例については、次の手順に従います。

1. Silverlight プロジェクトで、Visual Studio ツールボックスの[C1Image]アイコンをダブルクリックして、**C1Image** コントロールを **MainPage.xaml** に追加します。XAML マークアップは次のようになります。

```
XAML
<UserControl x:Class="C1Image.MainPage"
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
  mc:Ignorable="d"
  d:DesignHeight="300" d:DesignWidth="400"
  xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml">

  <Grid x:Name="LayoutRoot" Background="White">
    <c1:C1Image HorizontalAlignment="Left" Margin="10,10,0,0"
  Name="c1Image1" VerticalAlignment="Top" />
  </Grid>
</UserControl>
```

2. **C1Image** コントロールを選択し、**[プロパティ]** ウィンドウで、**Source** プロパティの横にある省略符ボタンをクリックします。**[画像の選択]** ダイアログボックスが開きます。
3. **[追加]** ボタンをクリックします。
4. **[開く]** ダイアログボックスで、アニメーション .gif を参照して見つけます。
5. 画像を選択し、**[開く]** をクリックします。
6. **[OK]** をクリックします。必要に応じて、画像のサイズと配置を調整できます。
7. ツールボックスで、**[共通 Silverlight コントロール]** の下にある **[CheckBox]** アイコンをダブルクリックします。
8. XAML マークアップで、**Content** を **Play**、**HorizontalAlignment** を **Center**、および **VerticalAlignment** を **Bottom** に設定します。XAML は、次のようになります。

```
XAML
<Grid x:Name="LayoutRoot" Background="White" Height="139" Width="384">
  <c1:C1Image HorizontalAlignment="Center" Margin="10,10,0,252"
  Name="c1Image1" Source="Images/Butterfly.gif" Width="44" />
  <CheckBox Content="再生" Height="16" HorizontalAlignment="Center"
  Margin="10,10,0,0" Name="checkBox1" VerticalAlignment="Bottom" />
</Grid>
```

9. MainPage.xaml.cs を開きます。
10. 次の **using** ステートメントを追加します (Visual Basic を使用する場合は **Imports**)。

```
C#
using C1.Silverlight.Imaging;
using C1.Silverlight;
```

11. **Play** メソッドと **Stop** メソッドのコードを追加します。次のようになります。

```
C#
public MainPage ()
{
  InitializeComponent ();
}
```

```
var gifImage = new ClGifImage(new Uri("/Images/Butterfly.gif",
UriKind.Relative));
c1Image1.Source = gifImage;

checkBox1.IsChecked = true;
checkBox1.Checked += delegate { gifImage.Play(); };
checkBox1.Unchecked += delegate { gifImage.Stop(); };
}
```

12. **[デバッグ]**→**[デバッグ開始]**をクリックして、アプリケーションを実行します。
13. アニメーショングラフィックを再生または停止するには、**[再生]**チェックボックスをオンまたはオフにします。

