Zip for UWP

2018.03.07 更新

グレープシティ株式会社

Zip for UWP

目次

Zip for UWP

Zip for UWP は、Zip 圧縮標準を完全に実装しています。ネットワークを介して送信されるデータを圧縮し、データの暗号化によってデータを保護することで、Windows ストアアプリケーションのパフォーマンスを向上させることができます。**C1Zip** ライブラリは、唯一の完全なクロスプラットフォーム Zip の実装です。

主な特長

Zip for UWP の主な機能の一部は以下のとおりです。

- **ファイルとフォルダの圧縮**
  圧縮ファイルは、占有するストレージ領域を削減し、非圧縮ファイルより高速に Web 経由で転送できます。リソースを圧縮ファイルとして保存し、クライアント側でリソースを効率的にダウンロードして圧縮解除することができます。また、複数のファイルを1つの圧縮フォルダにまとめると、ファイル群をより簡単に共有することができます。フォルダ構造を維持したまま、フォルダを圧縮したり展開することができます。
- **ファイル情報の取得と設定**
  zip ファイルのコンテンツの詳細なリストを初めとする zip ファイルの情報を簡単に取得できます。また、コメントの追加および取得によるファイル情報の制御、zip ファイル内の個々のエントリのパス情報の制御、zip ファイル全体のコメントの取得および設定を行うことができます。**C1ZipFile** クラスと **C1ZipEntry** クラスを使用して、インデックスまたはファイル名で zip アーカイブ内のファイルに簡単にアクセスできます。
- **データの暗号化**
  C1Zip はデータの暗号化をサポートしており、展開するにはパスワードが必要になる安全なアーカイブを作成できます。Zip の作成時に使用された元のパスワードを使用すれば、C1Zip を使用して、暗号化されているアーカイブを圧縮解除することもできます。
- **XML データに最適**
  XML は Web アプリケーションで最もよく使用される形式の1つです。通常、XML データは元のサイズの約 10 % に圧縮されます。
- **クロスプラットフォームの互換性**
  C1Zip ライブラリは、Windows フォーム、ASP.NET、WPF、Silverlight、Windows Phone、WinRT などのすべての .NET プラットフォームで利用できます。同じコードの大部分を再利用し、複数のプラットフォームをサポートすることができます。Microsoft ZipArchive クラスと Compression クラスは、Windows 8 の .NET 4.5 でしか利用できません。
- **複数のファイルおよびフォルダの圧縮**
  Zip for UWP を使用して、フォルダ構造を維持したまま、複数のフォルダをまとめて圧縮したり、zip ファイルからファイルを抽出することができます。

Zip の基本

C1.Xaml.Zip.dll には、データ圧縮サービスを提供するクラスが含まれます。Windows ストアアプリケーションでは、クライアントとサーバーの間でデータを転送する際にデータ圧縮は特に便利です。これらのクラスは、次の3つのレベルに分類されます。

レベル	主要なクラス	説明
高	**C1ZipFile**、**C1ZipEntry**、**C1ZipEntryCollection**	このレベルのクラスを使用して、zip ファイルを作成および管理したり、zip ファイルを開くことができます。また、zip ファイルの内容を調べたり、その整合性をテストすることができるほか、zip ファイルのエントリを追加、削除、および抽出することもできます。
中	**C1ZStreamReader**、**C1ZStreamWriter**	このレベルのクラスを使用して、通常の .NET ストリーム(メモリ、ファイル、ネットワークストリームを含む)にデータを圧縮したり、.NET ストリームからデータを展開することができます。
低	**ZStream**	これは、C1Zip で最も低いレベルのクラスです。これは、Zlib(Jean-loup Gailly と Mark Adler による一般的なデータ圧縮ライブラリ)の 100 % C# 実装です。ZStream は、C1Zip のより高レベルのクラスで使用されます。

高レベル:C1ZipFile、C1ZipEntry、および C1ZipEntryCollection クラス

これらは、**C1Zip** ライブラリで最も高いレベルのクラスです。このレベルのクラスを使用して、zip ファイルを作成および管理することができます。zip ファイルを使用してアプリケーションデータを保存することには、次の利点があります。

- 複数のファイルを1つに集約することができるので、アプリケーションの展開が容易になります。
- データを圧縮することで、ディスク容量とネットワーク帯域幅を節約することができます。
- zip 形式は、一般的なアプリケーションの多くでサポートされているオープンスタンダードです。

C1ZipFile クラス

C1ZipFile クラスは、zip ファイルをカプセル化します。C1ZipFile オブジェクトを作成したら、そのオブジェクトを既存の zip ファイルにアタッチしたり、そのオブジェクトに基づいて空の zip ファイルを新しく作成することができます。

次のコードは、sources.zip という名前の zip ファイルを作成し、"cs" という拡張子を持つすべてのファイルをその zip ファイルに追加します。

▶ C# コードの書き方

C#

```
// zip ファイルと圧縮するファイルのパスを取得します
string path = Application.ExecutablePath;
int pos = path.IndexOf(@"\bin");
path = path.Substring(0, pos + 1);
// zip ファイルを作成します
C1ZipFile zip = new C1ZipFile();
zip.Create(path + "source.zip");
```

```
// cs 拡張子を持つすべてのファイルを zip ファイルに追加します
foreach (string fileName in Directory.GetFiles(path, "*.cs"))
    zip.Entries.Add(fileName);
// 結果を表示します
foreach (C1ZipEntry ze in zip.Entries)
{
    Console.WriteLine("{0} {1:#,##0} {2:#,##0}",
        ze.FileName, ze.SizeUncompressed, ze.SizeCompressed);
}
```

C1ZipEntryCollection クラス

zip ファイルを作成または開いた後、**Entries** コレクションを使用して、zip ファイルの内容を調べたり、エントリの追加、展開、および削除を行います。次に例を示します。

▶ Visual Basic コードの書き方

Visual Basic

```
myZip.Entries.Add(stream1, "MyData.txt")
myZip.Entries.Add(stream2, "MyData.xml")
Dim zipEntry As C1ZipEntry
For Each zipEntry In myZip.Entries
    Console.WriteLine(zipEntry.FileName)
Next zipEntry
```

▶ C# コードの書き方

C#

```
myZip.Entries.Add(stream1, "MyData.txt");
myZip.Entries.Add(stream2, "MyData.doc");
foreach (C1ZipEntry zipEntry in myZip.Entries)
    Debug.WriteLine(zipEntry.FileName);
```

C1ZipEntry クラス

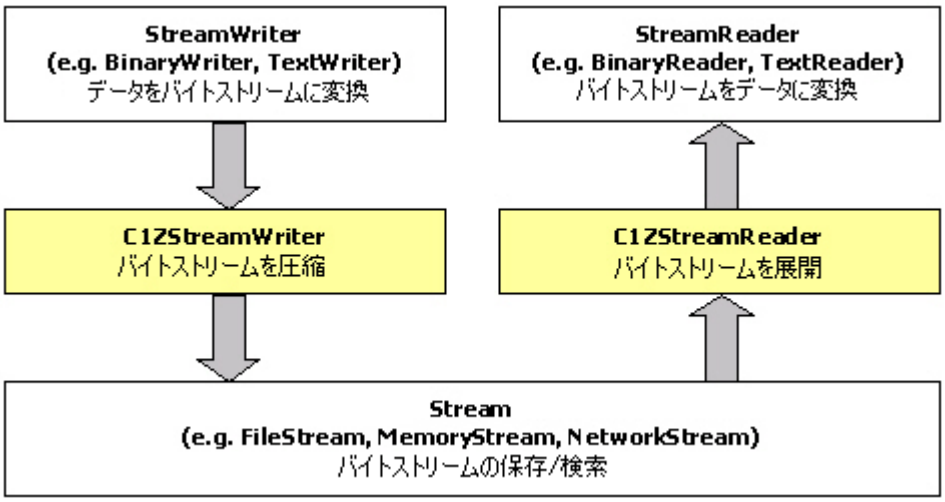
C1ZipEntry クラスは、各エントリについて説明するプロパティとメソッドを公開します。たとえば、エントリの元のファイル名、サイズ、圧縮サイズなどです。また、このクラスには、ストリームオブジェクトを返す **OpenReader** メソッドが含まれているため、エントリを展開しなくても内容を読み取ることができます。

中レベル：C1ZStreamReader および C1ZStreamWriter クラス

C1ZStreamReader クラスと **C1ZStreamWriter** クラスを使用することで、zip ファイル内だけでなく、任意の .NET ストリームでデータ圧縮を使用することができます。

C1ZStreamReader オブジェクトと C1ZStreamWriter オブジェクトを使用するには、それらを通常のストリームにアタッチし、そのストリームを通してデータを読み書きします。データは、基底のストリームで直ちに圧縮（または展開）されます。

この設計により、ネイティブ .NET ストリームとの統合性が高まります。次の図に、この設計がどのように機能するかを示します。



たとえば、次のコードを実行すると、DataGrid がストリームに保存されて読み取られます。

▶ C# コードの書き方

C#

```
PersonList personList = new PersonList();

        public MainPage()
        {
            InitializeComponent();
        }
        // DataGrid を作成します
        private void btnCreate_Click(object sender, RoutedEventArgs e)
        {
            for (int i = 0; i < 1000; i++)
            {
                personList.Persons.Add(new Person()
                {
                    FirstName = string.Format("First Name {0}", i),
                    LastName = string.Format("Last Name {0}", i),
                    Age = i,
                    City = string.Format("City {0}", i)
                });
            }

            this.dataGrid1.ItemsSource = personList.Persons;
        }

        // ストリームにデータを保存します
        private void btnSave_Click(object sender, RoutedEventArgs e)
        {
            SaveFileDialog dlgSaveFile = new SaveFileDialog();

            if (dlgSaveFile.ShowDialog() == true)
            {
                Stream stream = dlgSaveFile.OpenFile();

                DataContractSerializer dcs = new
DataContractSerializer(typeof(PersonList));
```

```
                dcs.WriteObject(stream, personList);
                stream.Close();
            }




        }
       // データを圧縮します
        private void btnSaveCompressed_Click(object sender, RoutedEventArgs e)
        {
            var dlgSaveFile = new SaveFileDialog();

            if (dlgSaveFile.ShowDialog() == true)
            {
                Stream stream = dlgSaveFile.OpenFile();
                //C1ZStreamWriter を使用してストリームを圧縮します。
                C1ZStreamWriter compressor = new C1ZStreamWriter(stream);

                DataContractSerializer dcs = new
DataContractSerializer(typeof(PersonList));
                dcs.WriteObject(compressor, personList);
                stream.Close();
            }
        }

      // ストリームからデータをロードします
        private void btnLoad_Click(object sender, RoutedEventArgs e)
        {
            var dlgOpenFile = new OpenFileDialog();
            this.dataGrid1.ItemsSource = null;

            if (dlgOpenFile.ShowDialog() == true)
            {
                Stream stream = dlgOpenFile.File.OpenRead();

                DataContractSerializer dcs = new
DataContractSerializer(typeof(PersonList));
                PersonList pl = (PersonList)dcs.ReadObject(stream);
                stream.Close();

                this.dataGrid1.ItemsSource = pl.Persons;

            }
        }

      // 圧縮データをロードします
        private void btnLoadCompressed_Click(object sender, RoutedEventArgs e)
        {
            var dlgOpenFile = new OpenFileDialog();
            this.dataGrid1.ItemsSource = null;
```

```
            if (dlgOpenFile.ShowDialog() == true)
            {
                Stream stream = dlgOpenFile.File.OpenRead();

                DataContractSerializer dcs = new
DataContractSerializer(typeof(PersonList));
                      //C1ZStreamReader を使用してストリームを圧縮解除します。
                C1ZStreamReader compressor = new C1ZStreamReader(stream);

                PersonList pl = (PersonList)dcs.ReadObject(compressor);
                stream.Close();

                this.dataGrid1.ItemsSource = pl.Persons;
            }
        }
```

低レベル:ZStream クラス

これは、**C1Zip** ライブラリ内で最も低いレベルのクラスで、前述のより高レベルのクラスで広く使用されます。

通常、直接 **ZStream** を使用する必要はありません。これは最も柔軟性の高いクラスですが、**C1Zip** ライブラリで最も使い方が難しいコンポーネントです。ZStream は、ZLIB ライブラリの C# 実装です。ZLIB は、Jean-loup Gailly と Mark Adler によるオープンソースライブラリです。

ZLIB の詳細については、http://www.info-zip.org/ または http://www.gzip.org/ を参照してください。

よくある質問

次に、**Zip for UWP** に関してよくある質問（FAQ）の一部を示します。

zip ファイルコメントに保存可能なデータ量を教えてください。zip ファイルコメントを使用して XML に情報を保存したいのです。

データ量は 32 KB に制限されています。この上限値に近い場合は、その情報を別のファイルとして追加することをお勧めします。

zip ファイルには最大何個のファイルを格納できますか。

同様に、32 k のエントリに制限されます。zip ファイル全体のサイズは4 GB に制限されています。これらの制限はすべて、zip ファイル仕様で使用されている変数の型に関係しています。興味深いことに、64 ビット拡張版の zip 形式に対する提案仕様がありますが、これはまだ広く使用されておらず、いまだ多くの議論があります。

タスク別ヘルプ

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

タスク別ヘルプの各トピックでは、具体的に C1.C1Zip 名前空間を参照するタスクを実行する方法について説明します。また、各トピックは、新しい Windows ストア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。

複数のファイルおよびフォルダの zip 圧縮

C1ZipEntryCollection の **AddFolderAsync** メソッドを使用して、複数のファイルやフォルダを簡単に zip 圧縮することができます。複数のファイルやフォルダを選択し、それらすべてをまとめて圧縮するには、次のコードを使用します。このコード例では、以下のボタンクリックイベントを定義しています。

- **_btnPickFolder_Click**：圧縮する複数のフォルダを選択します。
- **_btnPickFiles_Click**：圧縮するファイルを選択します。
- **_btnCompress_Click**：選択されたファイルやフォルダを 1 つのフォルダに圧縮開始します。
- **_btnExtract_Click**：zip 圧縮されたファイルやフォルダを抽出します。

次の手順を実行します。

1. **MainPage.xaml** ファイルを開き、開始タグ <Page> を見つけます。このタグには、必要な名前空間が含まれています。次のマークアップになるように、タグを編集します。

C#

```
<Page
    x:Class="ZipUWP.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:ZipUWP"
    xmlns:flexGrid="using:C1.Xaml.FlexGrid"
    xmlns:c1="using:C1.Xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">
```

2. 複数のファイルやフォルダを選択し、それらを 1 つの zip ファイルに圧縮し、最後にそれらを抽出するには、次のコードを追加します。

C#

```
    <Page.Resources>
        <ResourceDictionary>
            <ResourceDictionary.ThemeDictionaries>
                <ResourceDictionary x:Key="Default">
                    <SolidColorBrush x:Key="ApplicationPageBackgroundThemeBrush"
Color="#FF1D1D1D"/>
                </ResourceDictionary>
                <ResourceDictionary x:Key="Light">
                    <SolidColorBrush x:Key="ApplicationPageBackgroundThemeBrush"
Color="#FFF2F2F2"/>
                </ResourceDictionary>
            </ResourceDictionary.ThemeDictionaries>
            <Style x:Key="bigButton" TargetType="Button" >
                <Setter Property="Width" Value="124" />
                <Setter Property="FontSize"  Value="16" />
```

```
                <Setter Property="Margin" Value="10,0,0,0" />
                <Setter Property="Padding" Value="2" />
            </Style>
            <Style x:Key="smallButton" TargetType="Button" >
                <Setter Property="Width" Value="100" />
                <Setter Property="FontSize" Value="14" />
                <Setter Property="Margin" Value="4,0,0,0" />
                <Setter Property="Padding" Value="0" />
            </Style>
        </ResourceDictionary>
    </Page.Resources>
    <Grid>
        <Grid x:Name="_mainpage">
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto"/>
                <RowDefinition Height="Auto"/>
                <RowDefinition/>
            </Grid.RowDefinitions>
            <RelativePanel Grid.Row="0">
                <RelativePanel Name="_compressPanel" >
                    <TextBlock Name="_compress" Text="compress" FontSize="16"
Width="100"/>
                    <Button Name="_btnPickSingleFolder"
Content="PickSingleFolder" Click="_btnPickFolder_Click"/>
                    <Button Name="_btnPickFiles" Content="PickFiles"
RelativePanel.RightOf="_btnPickSingleFolder"
RelativePanel.AlignTopWith="_btnPickSingleFolder" Click="_btnPickFiles_Click"/>
                    <Button Name="_btnCompress" Content="Compress"
RelativePanel.RightOf="_btnPickFiles"
RelativePanel.AlignTopWith="_btnPickSingleFolder" Click="_btnCompress_Click"
IsEnabled="False" Height="51" Margin="4,0,-36,-47" Width="136"/>
                </RelativePanel>
                <RelativePanel Name="_extractPanel"  >
                    <TextBlock Name="_extract" Text="extract" FontSize="16"
Width="100"/>
                    <Button Name="_btnOpen" Content="Open"
Click="_btnOpen_Click" />
                    <Button Name="_btnExtract" Content="Extract"
RelativePanel.RightOf="_btnOpen" RelativePanel.AlignTopWith="_btnOpen"
Click="_btnExtract_Click" IsEnabled="False"/>
                </RelativePanel>
                <RelativePanel Name="_actionPanel" >
                    <Button Name="_btnRemove" Content="Remove"
Click="_btnRemove_Click" IsEnabled="False" />
                    <Button Name="_btnView" Content="View"
Click="_btnView_Click" IsEnabled="False"/>
                    <Button Name="_btnClear" Content="Clear"
Click="_btnClear_Click" />

                    <ListBox Name="listBox1" />
                </RelativePanel>
            </RelativePanel>
            <ProgressBar x:Name="progressBar" Grid.Row="1" Width="400"
```

```
Canvas.ZIndex="10" HorizontalAlignment="Center" VerticalAlignment="Center"
IsIndeterminate="True" Visibility="Collapsed" Background="Black"/>
            <flexGrid:C1FlexGrid BorderBrush="Gray" BorderThickness="1"
Margin="4"
                                 x:Name="_flex" Grid.Row="2"
                                 SelectionMode="Row"
                                 HeadersVisibility="Column"
                                 AllowResizing="Columns"
                                 HeaderGridLinesBrush="Transparent"
                                 AutoGenerateColumns="False" >
                <flexGrid:C1FlexGrid.Columns>
                    <flexGrid:Column Binding="{Binding FileName}" Header="Name"
/> <!--x:Uid="/ZipSamplesLib/Resources/Name"-->
                    <flexGrid:Column Binding="{Binding SizeUncompressedLong}"
Header="Size" Format="n0" /> <!--x:Uid="/ZipSamplesLib/Resources/Size"-->
                    <flexGrid:Column Binding="{Binding SizeCompressedLong}"
Header="Compressed" Format="n0" /> <!--
x:Uid="/ZipSamplesLib/Resources/Compressed"-->
                    <flexGrid:Column Binding="{Binding CompressionRatio}"
Header="Ratio" Format="p0" /> <!--x:Uid="/ZipSamplesLib/Resources/Ratio"-->
                    <flexGrid:Column Binding="{Binding CRC32}" Header="Crc"
Format="x0" /> <!--x:Uid="/ZipSamplesLib/Resources/CRC"-->
                    <flexGrid:Column Binding="{Binding Attributes}"
Header="Attributes" /> <!--x:Uid="/ZipSamplesLib/Resources/Attributes"-->
                </flexGrid:C1FlexGrid.Columns>
            </flexGrid:C1FlexGrid>
        </Grid>
        <RelativePanel Name="_preview" Visibility="Collapsed">
            <TextBlock Name="_text" Text="text" FontSize="28" Margin="0 0 40
0"/>
            <Button  Name="_btnClosePreview" Content="ClosePreview"
Click="_btnClosePreview_Click_1" RelativePanel.RightOf="_text"
Margin="30,0,40,0" Width="100"/>
            <TextBox Name="_tbContent" IsReadOnly="True" AcceptsReturn="True"
FontFamily="Courier New" Background="White"
RelativePanel.Below="_btnClosePreview" Margin="10"/>
        </RelativePanel>
        <VisualStateManager.VisualStateGroups>
            <VisualStateGroup x:Name="WindowSizeStates">

                <VisualState>
                    <VisualState.StateTriggers>

                        <AdaptiveTrigger MinWindowWidth="481" />
                    </VisualState.StateTriggers>
                    <VisualState.Setters>
                        <!-- Simplified Setter statements are used to move
elements around to optimize for more available space -->
                        <Setter Target="_btnPickSingleFolder.
(RelativePanel.RightOf)" Value="_compress" />
                        <Setter Target="_btnPickSingleFolder.Style" Value="
{StaticResource bigButton}" />
```

```
                        <Setter Target="_btnPickFiles.(RelativePanel.RightOf)"
Value="_btnPickSingleFolder" />
                        <Setter Target="_btnPickFiles.Style" Value="
{StaticResource bigButton}" />
                        <Setter Target="_btnCompress.(RelativePanel.RightOf)"
Value="_btnPickFiles" />
                        <Setter Target="_btnCompress.Style" Value="
{StaticResource bigButton}" />
                        <Setter Target="_extractPanel.(RelativePanel.Below)"
Value="_compressPanel" />
                        <Setter Target="_extractPanel.Margin" Value="0,10,0,0"
/>
                        <Setter Target="_btnOpen.(RelativePanel.RightOf)"
Value="_extract" />
                        <Setter Target="_btnOpen.Style" Value="{StaticResource
bigButton}" />
                        <Setter Target="_btnExtract.Style" Value="
{StaticResource bigButton}" />
                        <Setter Target="_actionPanel.(RelativePanel.Below)"
Value="_extractPanel" />
                        <Setter Target="_actionPanel.Margin" Value="100,10,0,0"
/>
                        <Setter Target="_btnRemove.Style" Value="{StaticResource
bigButton}" />
                        <Setter Target="_btnView.(RelativePanel.RightOf)"
Value="_btnRemove" />
                        <Setter Target="_btnView.Style" Value="{StaticResource
bigButton}" />
                        <Setter Target="_btnClear.(RelativePanel.RightOf)"
Value="_btnView" />
                        <Setter Target="_btnClear.Style" Value="{StaticResource
bigButton}" />
                        <Setter Target="_flex.FontSize" Value="16" />
                    </VisualState.Setters>
                </VisualState>
                <VisualState>
                    <VisualState.StateTriggers>

                        <AdaptiveTrigger MinWindowWidth="0" />
                    </VisualState.StateTriggers>
                    <VisualState.Setters>

                        <Setter Target="_compress.FontSize" Value="12" />
                        <Setter Target="_extract.FontSize" Value="12" />
                        <Setter Target="_btnPickSingleFolder.
(RelativePanel.Below)" Value="_compress" />
                        <Setter Target="_btnPickSingleFolder.Style" Value="
{StaticResource smallButton}" />
                        <Setter Target="_btnPickFiles.(RelativePanel.RightOf)"
Value="_btnPickSingleFolder" />
                        <Setter Target="_btnPickFiles.Style" Value="
{StaticResource smallButton}" />
                        <Setter Target="_btnCompress.(RelativePanel.RightOf)"
```

```
Value="_btnPickFiles" />
                        <Setter Target="_btnCompress.Style" Value="
{StaticResource smallButton}" />
                        <Setter Target="_extractPanel.(RelativePanel.Below)"
Value="_compressPanel" />
                        <Setter Target="_extractPanel.Margin" Value="0,10,0,0"
/>
                        <Setter Target="_btnOpen.(RelativePanel.Below)"
Value="_extract" />
                        <Setter Target="_btnOpen.Style" Value="{StaticResource
smallButton}" />
                        <Setter Target="_btnExtract.Style" Value="
{StaticResource smallButton}" />
                        <Setter Target="_actionPanel.(RelativePanel.Below)"
Value="_extractPanel" />
                        <Setter Target="_actionPanel.Margin" Value="0,10,0,0" />
                        <Setter Target="_btnRemove.Style" Value="{StaticResource
smallButton}" />
                        <Setter Target="_btnView.(RelativePanel.RightOf)"
Value="_btnRemove" />
                        <Setter Target="_btnView.Style" Value="{StaticResource
smallButton}" />
                        <Setter Target="_btnClear.(RelativePanel.RightOf)"
Value="_btnView" />
                        <Setter Target="_btnClear.Style" Value="{StaticResource
smallButton}" />
                        <Setter Target="_flex.FontSize" Value="10" />
                        <Setter Target="_btnClosePreview.Margin" Value="0,0,5,0"
/>
                        <Setter Target="_btnClosePreview.Width" Value="80" />
                    </VisualState.Setters>
                </VisualState>
            </VisualStateGroup>
        </VisualStateManager.VisualStateGroups>
    </Grid>
</Page>
```

3. **MainPage.xaml.cs** ファイルを開き、以下の名前空間をコードの先頭に追加します。

C#

```
using System.IO;
using C1.C1Zip;
using Windows.Storage.Pickers;
using Windows.Storage;
using Windows.UI.Popups;
```

4. 次のコードをファイルに追加します。このコードは、複数のファイルやフォルダを選択し、それらを1つのzip ファイルに圧縮し、最後にそれらを抽出します。

C#

```
namespace ZipUWP
{
    public sealed partial class MainPage : Page
    {
        C1ZipFile _zip;
```

```
        CollectionViewSource _cvs = new CollectionViewSource();
        MemoryStream zipMemoryStream = null;
        public MainPage()
        {
            this.InitializeComponent();
            _flex.SelectedItemChanged += _flex_SelectedItemChanged;
            // グリッドを entries コレクションに連結します
        }
        void _flex_SelectedItemChanged(object sender, EventArgs e)
        {
            if (_flex.SelectedItem != null)
            {
                _btnView.IsEnabled = true;
                _btnRemove.IsEnabled = true;
            }
            else
            {
                _btnView.IsEnabled = false;
                _btnRemove.IsEnabled = false;
            }
        }
        void RefreshView()
        {
            //var sel = _flex.SelectedItem;
            _flex.ItemsSource = null;
            if (_zip == null)
            {
                return;
            }
            _flex.ItemsSource = _zip.Entries;
            if (_zip.Entries.Count == 0)
            {
                _btnCompress.IsEnabled = false;
                _btnRemove.IsEnabled = false;
                _btnView.IsEnabled = false;
                _btnExtract.IsEnabled = false;
            }
            //_flex.SelectedItem = sel;
        }
        // 既存の zip ファイルを開きます
        async void _btnOpen_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var picker = new Windows.Storage.Pickers.FileOpenPicker();
                picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
                picker.FileTypeFilter.Add(".zip");
                StorageFile _zipfile = await picker.PickSingleFileAsync();
                if (_zipfile != null)
                {
                    Clear();
                    progressBar.Visibility = Visibility.Visible;
```

```
                    if (_zip == null)
                    {
                        _zip = new C1ZipFile(new System.IO.MemoryStream(),
true);
                    }
                    var stream = await
_zipfile.OpenAsync(Windows.Storage.FileAccessMode.ReadWrite);
                    _zip.Open(stream.AsStream());
                    _btnExtract.IsEnabled = true;
                    RefreshView();
                }
            }
            catch (Exception x)
            {
                System.Diagnostics.Debug.WriteLine(x.Message);
            }
            progressBar.Visibility = Visibility.Collapsed;
        }
        // 選択されたエントリを zip から削除します
        void _btnRemove_Click(object sender, RoutedEventArgs e)
        {
            foreach (C1ZipEntry entry in _flex.SelectedItems)
            {
                _zip.Entries.Remove(entry.FileName);
            }
            RefreshView();
        }
        // 選択されたエントリのプレビューを表示します
        void _btnView_Click(object sender, RoutedEventArgs e)
        {
            var entry = _flex.SelectedItem as C1ZipEntry;
            if (entry != null)
            {
                using (var stream = entry.OpenReader())
                {
                    var sr = new System.IO.StreamReader(stream);
                    _tbContent.Text = sr.ReadToEnd();
                }
                _preview.Visibility = Visibility.Visible;
                _mainpage.Visibility = Visibility.Collapsed;
            }
        }
        // プレビューペインを非表示にして閉じます
        void _btnClosePreview_Click_1(object sender, RoutedEventArgs e)
        {
            _preview.Visibility = Visibility.Collapsed;
            _mainpage.Visibility = Visibility.Visible;
        }
        // フォルダを指定してファイルを追加します
        private async void _btnPickFolder_Click(object sender, RoutedEventArgs
e)
        {
```

```
            try
            {
                FolderPicker folderPicker = new FolderPicker();
                folderPicker.FileTypeFilter.Add("*");
                StorageFolder pickedFolder = await
folderPicker.PickSingleFolderAsync();
                if (pickedFolder != null)
                {
                    if (_btnExtract.IsEnabled)
                    {
                        Clear();
                    }
                    progressBar.Visibility = Visibility.Visible;
                    if (zipMemoryStream == null)
                    {
                        zipMemoryStream = new MemoryStream();
                    }
                    if (_zip == null)
                    {
                        _zip = new C1ZipFile(zipMemoryStream, true);
                    }
                    await _zip.Entries.AddFolderAsync(pickedFolder);
                    _btnCompress.IsEnabled = true;
                }
            }
            catch
            {
            }
            RefreshView();
            progressBar.Visibility = Visibility.Collapsed;
        }
        // ファイルを追加します
        private async void _btnPickFiles_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var picker = new Windows.Storage.Pickers.FileOpenPicker();
                picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
                picker.FileTypeFilter.Add("*");
                var files = await picker.PickMultipleFilesAsync();
                if (files != null)
                {
                    if (files.Count == 0)
                    {
                        return;
                    }
                    if (_btnExtract.IsEnabled)
                    {
                        Clear();
                    }
                    progressBar.Visibility = Visibility.Visible;
                    if (zipMemoryStream == null)
```

```
                    {
                        zipMemoryStream = new MemoryStream();
                    }
                    if (_zip == null)
                    {
                        _zip = new C1ZipFile(zipMemoryStream, true);
                    }
                    foreach (var f in files)
                    {
                        await _zip.Entries.AddAsync(f);
                    }
                    _btnCompress.IsEnabled = true;
                }
            }
            catch
            {
            }
            RefreshView();
            progressBar.Visibility = Visibility.Collapsed;
        }
        private async void _btnCompress_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                if (zipMemoryStream != null)
                {
                    FileSavePicker fileSavePicker = new FileSavePicker();
                    fileSavePicker.FileTypeChoices.Add("Zip", new List<string> {
".zip" });
                    fileSavePicker.DefaultFileExtension = ".zip";
                    fileSavePicker.SuggestedFileName = "NewFolder";
                    fileSavePicker.CommitButtonText = "Save";
                    fileSavePicker.SuggestedStartLocation =
PickerLocationId.ComputerFolder;
                    StorageFile pickedSaveFile = await
fileSavePicker.PickSaveFileAsync();
                    await FileIO.WriteBytesAsync(pickedSaveFile,
zipMemoryStream.ToArray());
                    MessageDialog md = new MessageDialog("正常に圧縮し、パスは :" +
pickedSaveFile.Path);
                    md.ShowAsync();
                }
            }
            catch
            {
            }
            RefreshView();
        }
        private async void _btnExtract_Click(object sender, RoutedEventArgs e)
        {
            try
            {
```

```
                FolderPicker folderPicker = new FolderPicker();
                folderPicker.FileTypeFilter.Add("*");
                StorageFolder pickedFolder = await
folderPicker.PickSingleFolderAsync();
                progressBar.Visibility = Visibility.Visible;
                foreach (var entry in _zip.Entries)
                {
                    var name = entry.FileName;
                    await entry.Extract(pickedFolder, name);
                }
                MessageDialog md = new MessageDialog("正常に抽出し、パスは :" +
pickedFolder.Path);
                md.ShowAsync();
            }
            catch
            {
            }
            RefreshView();
            progressBar.Visibility = Visibility.Collapsed;
        }
        private void _btnClear_Click(object sender, RoutedEventArgs e)
        {
            Clear();
        }
        private void Clear()
        {
            _flex.ItemsSource = null;
            if (zipMemoryStream != null)
            {
                zipMemoryStream.Flush();
                zipMemoryStream.Dispose();
            }
            zipMemoryStream = null;
            _btnCompress.IsEnabled = false;
            _btnExtract.IsEnabled = false;
            if (_zip != null)
            {
                _zip.Close();
                _zip = null;
            }
        }
    }
}
```

上のコードには、zip ファイルに空のフォルダを追加できないという制限があることに注意してください。

Zip エントリからメモリにファイルを抽出する

zip からメモリ変数（バイト配列など）にファイルを抽出するには、次の関数を使用します。最初に、次の Imports（Visual Basic）/using（C#）文をコードの先頭に追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
Imports C1.C1Zip and Imports System.IO
```

▶ C# コードの書き方

C#

```
using C1.C1Zip; and using System.IO
```

次のコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
Private Function GetDataFromZipFile(zipFileName As String, entryName As String) As
Byte()
            ' zip ファイルからエントリを取得します。
            Dim zip As New C1ZipFile()
            zip.Open(zipFileName)
            Dim ze As C1ZipEntry = zip.Entries(entryName)
            ' エントリデータをメモリストリームにコピーします。
            Dim ms As New MemoryStream()
            Dim buf(1000) As Byte
            Dim s As Stream = ze.OpenReader()
            Try
                While True
                    Dim read As Integer = s.Read(buf, 0, buf.Length)
                    If read = 0 Then
                        Exit While
                    End If
                    ms.Write(buf, 0, read)
                End While
            Finally
                s.Dispose()
            End Try
            s.Close()
            ' 結果を返します。
            Return ms.ToArray()
        End Function
```

▶ C# コードの書き方

C#

```
private byte[] GetDataFromZipFile(string zipFileName, string entryName)
        {
            // zip ファイルからエントリを取得します。
            C1ZipFile zip = new C1ZipFile();
            zip.Open(zipFileName);
            C1ZipEntry ze = zip.Entries[entryName];
            // エントリデータをメモリストリームにコピーします。
            MemoryStream ms = new MemoryStream();
            byte[] buf = new byte[1000];
            using (Stream s = ze.OpenReader())
```

```
            {
                for (;;)
                {
                    int read = s.Read(buf, 0, buf.Length);
                    if (read == 0) break;
                    ms.Write(buf, 0, read);
                }
            }
            // C# には上の using ステートメントがあるため、close を呼び出す
            // 必要はありませんが、VB では close が必要です。
            //s.Close();
            // 結果を返します。
           return ms.ToArray();
        }
```

StreamReader を使用して、圧縮されたファイルを読み取る

StreamReader を使用して、圧縮されたファイルを読み取るには、次のコードを追加します。最初に、次の Imports（Visual Basic）/using（C#）文をコードの先頭に追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
Imports C1.C1Zip and Imports System.IO
```

▶ C# コードの書き方

C#

```
using C1.C1Zip; and using System.IO
```

次のコードを追加します。

▶ Visual Basic コードの書き方

Visual Basic

```
' zip ファイルを開きます。
Dim zip As New C1ZipFile()
zip.Open("c:\temp\myzipfile.zip")
' 任意のエントリの入力ストリームを開きます。
Dim ze As C1ZipEntry = zip.Entries("someFile.cs")
Dim s As Stream = ze.OpenReader()
' ストリームの StreamReader を開きます。
Dim sr As New StreamReader(s)
' StreamReader を使用し、閉じます。
```

▶ C# コードの書き方

C#

```
// zip ファイルを開きます。
C1ZipFile zip = new C1ZipFile();
zip.Open(@"c:\temp\myzipfile.zip");
```

```
// 任意のエントリの入力ストリームを開きます。
C1ZipEntry ze = zip.Entries["someFile.cs"];
Stream s = ze.OpenReader();
// ストリームの StreamReader を開きます。
StreamReader sr = new StreamReader(s);
// StreamReader を使用し、閉じます。
```

zip ファイルから画像を取得する

この例では、2個のボタンと1個のリストボックスを使用して、.zip ファイルから画像を取得する方法を示します。

zip ファイルから画像を直接取得するために、最初に、いくつかの画像ファイルを1つの zip ファイルに圧縮するコードを追加します。この例では、**btnNew_Click** イベントにこのコードを追加します。ここで、ボタンがクリックされたときに画像の新しい .zip ファイルを作成します。

▶ C# コードの書き方

C#

```
private void btnNew_Click(object sender, RoutedEventArgs e)
                {
                    // ファイルを開くダイアログを表示します。
                    SaveFileDialog dlgSaveFile = new SaveFileDialog();
                    dlgSaveFile.Filter = "Zip Files (*.zip) | *.zip";
                    // zip ファイルを開きます。
                    try
                    {
                        if (dlgSaveFile.ShowDialog() == true)
                        {

        zipFile.Create(dlgSaveFile.OpenFile());
                        }
                    }
                    catch
                    {
                        MessageBox.Show("ZIP ファイルを作成できませんでした。もう一度作成してください。",
        "C1Zip", MessageBoxButton.OK);
                    }
                }
```

次に、次のコードを使用して **ListBox** 内のリストに画像ファイルを追加します。

▶ C# コードの書き方

C#

```
private void btnAdd_Click(object sender, RoutedEventArgs e)
        {
            // 追加するファイルのリストを取得します。
            OpenFileDialog fo = new OpenFileDialog();
            fo.Multiselect = true;
            if (fo.ShowDialog() == true)
            {
                // リストにファイルを追加します。
```

```
                foreach (FileInfo file in fo.Files)
                {
                    Stream stream = file.OpenRead();
                    listBox1.Items.Add(file.Name);
                    zipFile.Entries.Add(stream, file.Name);
                }
```

画像を選択できるようにするには、画像データを含むストリームを取得し（**OpenReader** メソッド）、次のコードを listBox1_SelectionChanged イベントと **StreamCopy** イベントに追加します。

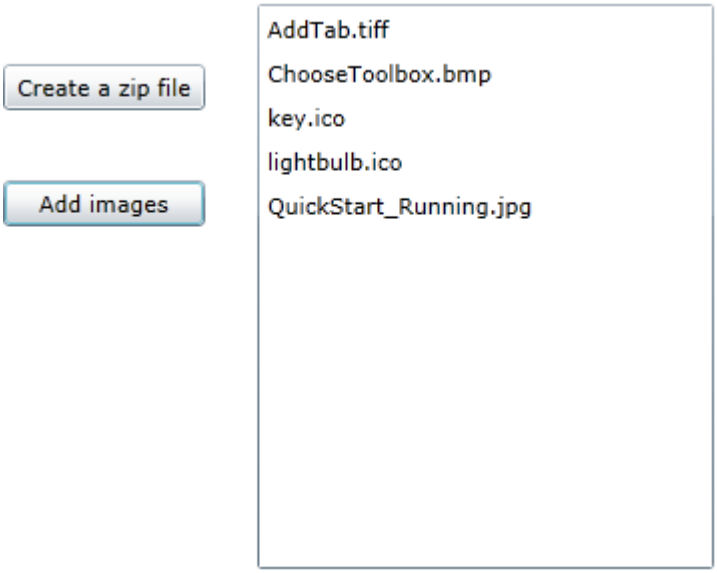
▶ C# コードの書き方

C#

```
private void listBox1_SelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            // 選択された項目を取得します。
            string item = (string)listBox1.SelectedItem;
            // 画像を圧縮ストリームから直接ロードします。
            Stream stream = zipFile.Entries[item].OpenReader();
            using (MemoryStream ms = new MemoryStream())
            {
                StreamCopy(ms, stream);
                BitmapImage img = new BitmapImage();
                img.SetSource(ms);
                this.image1.Source = img;
                // ストリームを閉じます。
                stream.Close();
            }
        }
        private void StreamCopy(Stream dstStream, Stream srcStream)
        {
            byte[] buffer = new byte[32768];
            for (; ; )
            {
                int read = srcStream.Read(buffer, 0, buffer.Length);
                if (read == 0) break;
                dstStream.Write(buffer, 0, read);
            }
            dstStream.Flush();
        }
```

このトピックの作業結果

ICO、TIFF、BMP、JPG などのいくつかのタイプの画像が表示されます。



文字列変数を zip ファイルに保存する

文字列変数を zip ファイルに保存するには、次のいずれかのメソッドを使用します。

- **C1ZipEntryCollection.OpenWriter** メソッド
  OpenWriter メソッドを使用して、ストリームライタを取得し、それに文字列を書き込み、最後に閉じます。データは、ストリームへの書き込み時に圧縮されます。ストリームを閉じると、ストリーム全体が zip ファイルに保存されます。
- **MemoryStream メソッド**
  MemoryStream メソッドを使用して、ストリームにデータを書き込み、zip ファイルに追加します。このメソッドでは、OpenWriter メソッドを使用する場合より少し多くの作業を行う必要があります。それでも、これは扱いやすいメソッドです。

次のコードは、両方のメソッドを示しています。この例では、**OpenWriter** メソッドのコードは **button1_Click** イベント内に示されます。**MemoryStream** メソッドのコードは button2_Click イベント内に示されます。

▶ C# コードの書き方

C#

```
 private void button1_Click(object sender, RoutedEventArgs e)
                {
                    SaveFileDialog dlgSaveFile = new SaveFileDialog();
                    dlgSaveFile.Filter = "Zip Files (*.zip) | *.zip";
                    if (dlgSaveFile.ShowDialog() == true)
                    {
                        zipFile.Create(dlgSaveFile.OpenFile());
                    }
                    // 方法1:OpenWriter を使用する。
                    Stream stream = zipFile.Entries.OpenWriter("Shakespeare.txt",
true);
                    C1ZStreamWriter sw = new C1ZStreamWriter(stream);
                    byte[] text =
System.Text.Encoding.Unicode.GetBytes(shakespeareText);
                    sw.Write(text, 0, text.Length);
                    sw.Flush();
                    stream.Close();
                }
```

```
                private void button2_Click(object sender, RoutedEventArgs e)
                {
                    SaveFileDialog dlgSaveFile = new SaveFileDialog();
                    dlgSaveFile.Filter = "Zip Files (*.zip) | *.zip";
                    if (dlgSaveFile.ShowDialog() == true)
                    {
                        zipFile.Create(dlgSaveFile.OpenFile());
                    }
                    // 方法2：メモリストリームを使用する。
                    Stream stream = new MemoryStream();
                    C1ZStreamWriter sw = new C1ZStreamWriter(stream);
                    byte[] text =
System.Text.Encoding.Unicode.GetBytes(shakespeareText);
                    sw.Write(text, 0, text.Length);
                    sw.Flush();
                    stream.Position = 0;
                    zipFile.Entries.Add(stream, "Shakespeare2.txt");
                    stream.Close();
                }
```

圧縮のレベルを設定する

圧縮ファイルのサイズを最小化するには、次のコードを使用して、C1ZStreamWriter のコンストラクタで圧縮レベルを設定します。

▶ C# コードの書き方

C#

```
SaveFileDialog dlgSaveFile = new SaveFileDialog();
                    if (dlgSaveFile.ShowDialog() == true)
                    {
                        C1ZStreamWriter compressor = new
        C1ZStreamWriter(dlgSaveFile.OpenFile(),

        CompressionLevelEnum.BestCompression);
                    }
```

このコードサンプルでは、圧縮レベルを **BestCompression** に設定しています。これは、圧縮時間が最長で、圧縮速度が最低になります。C1ZStreamWriter のコンストラクタでは、ほかに次の圧縮レベルオプションを指定できます。

- **BestSpeed** を指定すると、圧縮時間は最短、圧縮速度は最高になります。
- **DefaultCompression** を指定すると、圧縮時間と速度が中程度になります。
- **NoCompression** を指定すると、データは圧縮されません。

パスワードを使用して zip ファイル保護する

パスワードで保護された zip ファイルを作成するには、エントリを作成する前に、**Password** プロパティを空以外の文字列に設定します。エントリごとに独自のパスワードを設定できます。次に例を示します。

▶ Visual Basic コードの書き方

Visual Basic

```
Dim zip As New C1ZipFile()
zip.Password = "password"
zip.Entries.Add(someFile)
```

▶ C# コードの書き方

C#

```
C1ZipFile zip = new C1ZipFile();
zip.Password = "password";
zip.Entries.Add(someFile);
```

このエントリを後から抽出するには、エントリを追加するときに適用された値と同じ値を **Password** プロパティに設定する必要があります。次に例を示します。

▶ Visual Basic コードの書き方

Visual Basic

```
' 失敗します。パスワードが設定されていません。
zip.Password = ""
zip.Entries.Extract(someFile)
' 失敗します。パスワードが不正です。
zip.Password = "pass"
zip.Entries.Extract(someFile)
' 正しく実行されます。
zip.Password = "password"
zip.Entries.Extract(someFile)
```

▶ C# コードの書き方

C#

```
// 失敗します。パスワードが設定されていません。
zip.Password = "";
zip.Entries.Extract(someFile);
// 失敗します。パスワードが不正です。
zip.Password = "pass";
zip.Entries.Extract(someFile);
// 正しく実行されます。
zip.Password = "password";
zip.Entries.Extract(someFile);
```

埋め込みリソースから Zip ファイルを開く

次のコードを使用して、プロジェクトの埋め込みリソースからファイルをロードすることができます。この例では、プロジェクトの名前空間は "ZipFileDemo" です。Sample.zip ファイルはプロジェクトのルートディレクトリにあり、その Build プロパティは Embedded Resource に設定されています。

次の imports 文をプロジェクトの先頭に追加します。

▶ C# コードの書き方

C#

```
using System.Reflection;
```

```
using C1.C1Zip;
```

プロジェクトに次のコードを追加します。

▶ C# コードの書き方

C#

```
// 埋め込みリソースから zip ファイルを開きます
C1ZipFile zip = new C1ZipFile();
Assembly asm =typeof(MainPage).GetTypeInfo().Assembly;
Stream stream = asm.GetManifestResourceStream("ZipFileDemo.Sample.zip");
zip.Open(stream);
// FlexGrid にエントリをロードします
//_flex.ItemsSource = zip.Entries;
```

Web から Zip ファイルをロードする

Web から zip ファイルを非同期にダウンロードすることで、アプリケーションサイズを削減することができます。次のコードを使用して、Web にある zip ファイルを **C1ZipFile** オブジェクトで開きます。

▶ C# コードの書き方

C#

```
using C1.C1Zip;
using System.Net.Http;using Windows.UI.Popups;
private async void LoadZipFile()
{
    // Web からファイルをロードします
    HttpClient client = new HttpClient();
    C1ZipFile zip = new C1ZipFile();
    try
    {
        // zip をバイト配列にダウンロードします。
        var byteArray = await client.GetByteArrayAsync(new Uri("htt://yourfile.zip",
UriKind.Absolute));
        // ストリーミングして開くバイト配列を書き込みます
        MemoryStream ms = new MemoryStream();
        ms.Write(byteArray, 0, byteArray.Length);
        zip.Open(ms);
        // FlexGrid にエントリをロードします
        _flex.ItemsSource = zip.Entries;
    }
    catch (Exception ex)
    {
        var dialog = new MessageDialog(ex.Message);
        dialog.ShowAsync();
    }
}
```

ユーザーのコンピュータから Zip ファイルを開く

FileOpenPicker クラスを使用して、ユーザーがドキュメントライブラリから zip ファイルを選択して開くことができるようにします。次のコードを使用して、C1Zip でファイルを開きます。また、このサンプルは、結果を表示するために、ページに "flex" という名前の C1FlexGrid コントロールがあることを前提としています。ただし、そうしなければならないというわけではありません。

▶ C# コードの書き方

C#

```
// 既存の zip ファイルを開きます
async void _btnOpen_Click_1(object sender, RoutedEventArgs e)
{
    C1ZipFile zip = new C1ZipFile(new System.IO.MemoryStream(), true);
    try
    {
        var picker = new Windows.Storage.Pickers.FileOpenPicker();
        picker.SuggestedStartLocation =
Windows.Storage.Pickers.PickerLocationId.DocumentsLibrary;
        picker.FileTypeFilter.Add(".zip");
        var file = await picker.PickSingleFileAsync();
        if (file != null)
        {
            var stream = await
file.OpenAsync(Windows.Storage.FileAccessMode.ReadWrite);
            zip.Open(stream.AsStream());

            // FlexGrid にエントリをロードします
            flex.ItemsSource = zip.Entries;
        }
    }
    catch (Exception x)
    {
        System.Diagnostics.Debug.WriteLine(x.Message);
    }
}
```

Zip 内の特定のファイルにアクセスする

C1Zip では、Entries コレクションを使用して、Zip アーカイブ内の個々のファイルやフォルダにアクセスすることができます。次の例は、"_zip" という名前の C1ZipFile のインスタンスを使用して、zip ファイルが既に開かれていることを前提としています。

C#

```
// 任意のエントリの入力ストリームを開きます。
C1ZipEntry ze = _zip.Entries["someFile.cs"];
Stream s = ze.OpenReader();
// ストリームの StreamReader を開きます。
StreamReader sr = new StreamReader(s);
// StreamReader を使用し、閉じます。
```