Uploader for Silverlight

2013.05.29 更新

グレープシティ株式会社

Uploader for Silverlight

目次

製品の概要

Uploader for Silverlight を使用すると、Silverlight アプリケーションからサーバーにファイルをアップロードできます。**Uploader for Silverlight** では、簡単かつ信頼性の高い方法で、ファイルやストリームをサーバーにアップロードできます。

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。

主な特長

Uploader for Silverlight には、次の主な特長があります。

- **ファイルサイズの検証**
  ファイルのアップロード要求に対する最大ファイルサイズを設定します。詳細については、「最大アップロードサイズ」を参照してください。
- **非同期ファイルアップロード**
  エンドユーザーは、FilePicker for Silverlight コントロールを使って複数のファイルを同時に選択し、**C1Uploader** コントロールを使ってこれらのファイルを非同期にアップロードできます。
- **マルチパート POST メッセージの使用**
  C1UploaderPost オブジェクトを使用すると、マルチパート POST メッセージを使ってファイルをサーバーにアップロードできます。
- **Web サービスの使用**
  C1UploaderWebService オブジェクトを使用すると、Web サービスを使ってファイルをサーバーにアップロードできます。
- **大きなファイルのアップロード**
  サーバー構成を変更しなくても、大きなファイルをアップロードできます。**Uploader for Silverlight** は、ストリームを小さく分割します。分割されたストリームは、サーバーで再度結合されます。
- **複数のソースからのファイルのアップロード**
  エンドユーザーのコンピュータで選択されたファイル、分離ストレージファイル、またはメモリ内のストリームからコンテンツをアップロードできます。

クイックスタート

このクイックスタートは、**Uploader 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、**C1Uploader** クラスをアプリケーションに追加します。次に、ファイルをアップロードできるようにアプリケーションを設定します。

手順 1:アプリケーションの作成

この手順では、最初に Visual Studio で **Uploader for Silverlight** を使用する Silverlight アプリケーションを作成します。新しい Silverlight プロジェクトを作成し、アプリケーションを設定します。

プロジェクトを設定するには、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の左ペインから言語を選択し、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**(たとえば、「QuickStart」)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。
3. **[OK]**をクリックしてデフォルト設定を受け入れ、**[新しい Silverlight アプリケーション]**ダイアログボックスを閉じると、プロジェクトが作成されます。**MainPage.xaml** ファイルが開きます。
4. ソリューションエクスプローラウィンドウでプロジェクトを右クリックし、**[参照の追加]**を選択します。
5. **[参照の追加]**ダイアログボックスで、**C1.Silverlight.dll** および **C1.Silverlight.Uploader.dll** アセンブリを見つけて選択し、 **[OK]** をクリックしてプロジェクトに参照を追加します。
6. プロジェクトの XAML ウィンドウで、カーソルを <Grid> タグと </Grid> タグの間に置き、1回クリックします。
7. 次の XAML マークアップを **MainPage.xaml** ファイルの<Grid> タグと </Grid>タグの間に追加します。

   XAML

   ```
   <StackPanel Orientation="Horizontal" Height="250" HorizontalAlignment="Center">
     <Button Content="画像のアップロード" Height="30" VerticalAlignment="Top"
   Margin="5" Click="Upload_Click" />
     <ScrollViewer Width="400" Margin="5" Background="White">
       <StackPanel x:Name="PicturePanel"/>
     </ScrollViewer>
   </StackPanel>
   ```

   このマークアップは、1つのボタンと1つのパネルを作成します。ユーザーは、このボタンをクリックして画像ファイルをサーバーにアップロードします。また、パネルには画像が表示されます。

8. ソリューションエクスプローラで **QuickStart.Web** アプリケーションを右クリックし、[追加]→**[新しいフォルダ]**を選択します。フォルダ名を "Pictures" と指定します。ただし、プロジェクトを "QuickStart" 以外の名前にした場合は、指定したプロジェクトの名前が "MyProject" だとして、**MyProject.Web** を右クリックします。

   このフォルダは、アップロードした画像を保存するために使用されます。Pictures フォルダを作成したら、アプリケーションがピクチャをそのフォルダに保存できるように、フォルダの権限を設定する必要があります。ネットワークサービスがこのフォルダに対する読み書き権限を持っていることを確認する必要があります。

ここまでの成果

Silverlight アプリケーションを正しく作成および設定し、コントロールをページに追加しました。次の手順では、**C1Uploader** クラスを実装するためのコードを追加します。

手順 2:コードの追加

前の手順で、Silverlight アプリケーションを設定しました。ただし、この時点でアプリケーションを実行しても、ボタンとパネルは動作しません。この手順では、引き続き Visual Studio で作業し、プロジェクトに **C1Uploader** クラスを実装するためのコードを追加します。

C1Uploader クラスを実装するには、次の手順に従います。

1. ソリューションエクスプローラに移動し、**MainPage.xaml** ファイルを右クリックして**[コードの表示]**を選択し、コードビューに切り替えます。
2. コードビューで、次の Imports または using 文をページの先頭に追加します(ページに含まれていない場合)。

VisualBasic

```
Imports System.Windows.Controls
Imports System.IO
Imports System.Diagnostics
Imports C1.Silverlight
Imports C1.Silverlight.Uploader
```

C#

```
using System.Windows.Controls;
using System.IO;
using System.Diagnostics;
using C1.Silverlight;
using C1.Silverlight.Uploader;
```

3. **MainPage.xaml.cs** または MainPage.xaml.vb ファイルで **MainPage** クラスの他のすべてのメソッドの下に、次のイベントハンドラを追加します。

VisualBasic

```
Private Sub Upload_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
    ' OpenFileDialog を設定します
    Dim dlg As New OpenFileDialog()
    dlg.Filter = "すべてのファイル|*.*|画像|*.jpg;*.png;*.gif"
    dlg.FilterIndex = 2
    dlg.Multiselect = True
    ' OpenFileDialog を表示します
    If CBool(dlg.ShowDialog()) Then
        Try
        ' ファイルが実際に画像かどうかをチェックします
        For Each fdi As FileInfo In dlg.Files
            Dim img As New System.Windows.Media.Imaging.BitmapImage()
            img.SetSource(fdi.OpenRead())
        Next
        Catch
```

```
   C1MessageBox.Show("少なくとも1個のファイルが画像ではないようです。", "アップロード中止",
C1MessageBoxButton.OK, C1MessageBoxIcon.Warning)
        Exit Sub
        End Try
        ' アップローダーを作成して設定します
         Dim uri As Uri = BuildAbsoluteUri("photoUpload.ashx")
        Dim uploader As C1Uploader = New C1UploaderPost(uri)
        AddHandler uploader.UploadProgressChanged, AddressOf
uploader_UploadProgressChanged
        AddHandler uploader.UploadCompleted, AddressOf uploader_UploadCompleted
        ' 選択されたファイルをアップロードします
         uploader.AddFiles(dlg.Files)
        uploader.BeginUploadFiles()
   End If
End Sub
```

C#

```
private void Upload_Click(object sender, RoutedEventArgs e)
{
    // OpenFileDialog を設定します
    OpenFileDialog dlg = new OpenFileDialog();
    dlg.Filter = "すべてのファイル|*.*|画像|*.jpg;*.png;*.gif"
    dlg.FilterIndex = 2;
    dlg.Multiselect = true;
    // OpenFileDialog を表示します
    if ((bool)dlg.ShowDialog())
    {
        try
        {
            // ファイルが実際に画像かどうかをチェックします
            foreach (FileInfo fdi in dlg.Files)
            {
              System.Windows.Media.Imaging.BitmapImage img = new
System.Windows.Media.Imaging.BitmapImage();
              img.SetSource(fdi.OpenRead());
            }
        }
        catch
        {
            C1MessageBox.Show(
            "少なくとも1個のファイルが画像ではないようです。",
            "アップロード中止", C1MessageBoxButton.OK, C1MessageBoxIcon.Warning);
            return;
        }
        // アップローダーを作成して設定します
        Uri uri = BuildAbsoluteUri("photoUpload.ashx");
        C1Uploader uploader = new C1UploaderPost(uri);
        uploader.UploadProgressChanged += uploader_UploadProgressChanged;
        uploader.UploadCompleted += uploader_UploadCompleted;
        // 選択されたファイルをアップロードします
```

```
            uploader.AddFiles(dlg.Files);
            uploader.BeginUploadFiles();
        }
    }
```

このイベントハンドラは、最初に、ユーザーがアップロードするファイルを選択するための **OpenFileDialog** を表示します。Silverlight は安全なサンドボックス環境で実行されるため、ファイルシステムに直接アクセスすることはできません。ただし、**OpenFileDialog** クラスは、ユーザーによって明示的に選択された特定のファイルへのアクセスを許可します。**IsolatedStorage** 領域内のファイルにアクセスすることもできます。

ファイルが選択されると、このコードは、それらのファイルが実際に画像かどうかをチェックします。それには、**BitmapImage** オブジェクトを作成し、それらのファイルをこのオブジェクトにロードします。ファイルに画像が含まれていない場合は、例外が生成され、処理が中断されます。このチェックにより、ユーザーが不正な種類のファイルを誤ってアップロードしたり、悪質な目的でアップロードすることを防止できます。後で、サーバー側でも同様のセキュリティチェックを実行します。

ファイルが検証されると、このコードは新しい **C1UploaderPost** オブジェクトを作成します。**C1Uploader** オブジェクトには、**C1UploaderPost** と **C1UploaderWebService** の2種類あります。この2つの違いは、サーバー側でファイルを受信する方法です。**C1UploaderPost** は、次の手順で示すように、とても使いやすいマルチパート POST プロトコルを使用します。

4. プロジェクトに次のイベントハンドラを追加します。

VisualBasic

```
Public Shared Function BuildAbsoluteUri(ByVal relativeUri As String) As Uri
    ' 現在の絶対 URI を取得します。これは、アプリケーションが展開される場所に依存します
    Dim uri As Uri = System.Windows.Browser.HtmlPage.Document.DocumentUri
    Dim uriString As String = uri.AbsoluteUri
    ' ページ名を相対サービス URI に置き換えます
    Dim ls As Integer = uriString.LastIndexOf("/"c)
    uriString = uriString.Substring(0, ls + 1) + relativeUri
    ' 新しい URI を返します
    Return New Uri(uriString, UriKind.Absolute)
End Function
```

C#

```
public static Uri BuildAbsoluteUri(string relativeUri)
{
    // 現在の絶対 URI を取得します。これは、アプリケーションが展開される場所に依存します
    Uri uri = System.Windows.Browser.HtmlPage.Document.DocumentUri;
    string uriString = uri.AbsoluteUri;
    // ページ名を相対サービス URI に置き換えます
    int ls = uriString.LastIndexOf('/');
    uriString = uriString.Substring(0, ls + 1) + relativeUri;
    // 新しい URI を返します
    return new Uri(uriString, UriKind.Absolute);
}
```

C1UploaderPost コンストラクタには、サーバー側でファイルを処理するハンドラの **Uri** を指定するパラメータが必要です。Silverlight は絶対 **Uri** を要求するため、このコードでは、アプリケーションが開発マシンで実行されていても、本

稼動サーバーで実行されていても機能するヘルパーメソッド **BuildAbsoluteUri** を使用します。

また、このコードは、イベントハンドラを新しい **C1UploaderPost** の UploadProgressChanged および UploadCompleted イベントにアタッチします。次の手順で追加するコードは、前者を使ってアップロード処理の進捗状況をチェックし、後者を使ってアップロードされたファイルを表示します。

5. プロジェクトに次のイベントハンドラを追加します。

VisualBasic

```
Private Sub uploader_UploadProgressChanged(ByVal sender As Object, ByVal e As
C1.Silverlight.Uploader.UploadProgressChangedEventArgs)
    Debug.WriteLine("アップロード中 {0}／{1} バイト、{2}%", e.BytesUploaded,
e.TotalBytesToUpload, e.ProgressPercentage)
End Sub
Private Sub uploader_UploadCompleted(ByVal sender As Object, ByVal e As
UploadCompletedEventArgs)
    If e.Success Then
        For Each fileName As String In e.Files
        ' ピクチャとタイトルのホルダ
        Dim picHolder As New StackPanel()
        picHolder.Width = 200
        ' ピクチャを表示するための Image を作成します
        Dim bmp As New System.Windows.Media.Imaging.BitmapImage()
        bmp.UriSource = BuildAbsoluteUri("Pictures/" & fileName)
        Dim img As New Image()
        img.Source = bmp
        img.Margin = New Thickness(10, 10, 10, 0)
        picHolder.Children.Add(img)
        ' 画像タイトルを作成します
        Dim tb As New TextBlock()
        tb.Text = fileName
        tb.FontSize = 12
        tb.HorizontalAlignment = HorizontalAlignment.Center
        tb.Margin = New Thickness(10, 0, 10, 10)
        picHolder.Children.Add(tb)
        ' パネルにピクチャホルダを追加します
        PicturePanel.Children.Add(picHolder)
        Next
    Else
        Debug.WriteLine("アップロード失敗: {0}", If(e.[Error] IsNot Nothing, e.
[Error].Message, "??"))
    End If
End Sub
```

C#

```
void uploader_UploadProgressChanged(object sender,
C1.Silverlight.Uploader.UploadProgressChangedEventArgs e)
{
    Debug.WriteLine("アップロード中 {0}／{1} バイト、{2}%",
```

```
    e.BytesUploaded,
    e.TotalBytesToUpload,
    e.ProgressPercentage);
}
void uploader_UploadCompleted(object sender, UploadCompletedEventArgs e)
{
    if (e.Success)
    {
        foreach (string fileName in e.Files)
        {
            // ピクチャとタイトルのホルダ
            StackPanel picHolder = new StackPanel();
           picHolder.Background = new SolidColorBrush(Colors.LightGray);
           picHolder.Margin = new Thickness(10);
           picHolder.Width = 200;
           // ピクチャを表示するための Image を作成します
            System.Windows.Media.Imaging.BitmapImage bmp = new
System.Windows.Media.Imaging.BitmapImage();
           bmp.UriSource = BuildAbsoluteUri("Pictures/" + fileName);
           Image img = new Image();
           img.Source = bmp;
           img.Margin = new Thickness(10, 10, 10, 0);
           picHolder.Children.Add(img);
           // 画像タイトルを作成します
            TextBlock tb = new TextBlock();
           tb.Text = fileName;
           tb.FontSize = 12;
           tb.HorizontalAlignment = HorizontalAlignment.Center;
           tb.Margin = new Thickness(10, 0, 10, 10);
           picHolder.Children.Add(tb);
           // パネルにピクチャホルダを追加します
            PicturePanel.Children.Add(picHolder);
        }
    }
    else
    {
        Debug.WriteLine("アップロード失敗: {0}", e.Error != null ? e.Error.Message :
"??");
    }
}
```

このコードは、アップロードが成功したかどうかをチェックし、次にファイルをループ処理して、ファイルごとに新しい **Image** 要素を作成します。**Image** 要素は、画像がサーバーの「Pictures」フォルダに保存されると仮定して、画像を表示するように設定されます。

ここまでの成果

Silverlight アプリケーションを正しく作成し、コントロールをページに追加しました。ここでプロジェクトを実行し、いくつかのピクチャをアップロードしてみてください。エラーが表示されます。**C1Uploader** はサーバーにファイルを送信しますが、まだサーバーコードが実装されていないためです。次の手順では、このサーバーコードを実装します。

手順 3：サーバーコードの追加

前の手順では、Silverlight アプリケーションを設定し、クライアントのアクションと動作を実装するコードをアプリケーションに追加しました。ただし、アプリケーションを実行すると、サーバーコードが実装されていないためにエラーが発生します。この手順では、引き続き Visual Studio で作業し、プロジェクトにサーバーコードを実装するためのコードを追加します。

サーバーコードを実装するには、次の手順に従います。

1. Visual Studio のソリューションエクスプローラで、**QuickStart.Web** ソリューションを右クリックし、**[追加]→[新しい項目]**オプションを選択します。ただし、プロジェクトを "QuickStart" 以外の名前にした場合は、指定したプロジェクトの名前が "MyProject" だとして、**MyProject.Web** を右クリックします。
2. **[新しい項目の追加]**ダイアログボックスで、**[ジェネリックハンドラー]**テンプレートを選択し、新しいクラス名を "photoUpload.ashx" に指定します。これは、前に **C1Uploader** オブジェクトを作成した際に使用した **URI** です。
3. **[追加]**をクリックしてプロジェクトにファイルを追加したら、**[新しい項目の追加]**ダイアログボックスを閉じます。
4. **photoUpload.ashx** ファイルを開き、**ProcessRequest** メソッドのデフォルトの実装を次のコードに置き換えます。

VisualBasic

```
Sub ProcessRequest(ByVal context As HttpContext) Implements
IHttpHandler.ProcessRequest
    Const MAXFILES As Integer = 30
    ' クリーンアップします
    Dim path As String = context.Request.MapPath("Pictures")
    Dim files As String() = System.IO.Directory.GetFiles(path)
    If files.Length >= MAXFILES Then
        For Each fileName As String In files
        System.IO.File.Delete(fileName)
        Next
    End If
    ' アップロードされるファイルを取得します
    Dim i As Integer = 0
    While i < context.Request.Files.Count AndAlso i < MAXFILES
        Dim file As HttpPostedFile = context.Request.Files(i)
        Try
        ' Image オブジェクトにファイルをロードして、ファイルが実際に画像かどうかをチェックします
        Dim img As System.Drawing.Image =
System.Drawing.Image.FromStream(file.InputStream)
        ' ファイルを Pictures フォルダに保存します
        Dim fileName As String = context.Request.MapPath("Pictures/" &
file.FileName)
        file.SaveAs(fileName)
        Catch
        ' 有効な画像でない場合は、要求をスキップします
        Exit Try
        End Try
        i += 1
    End While
End Sub
```

C#

```
public void ProcessRequest(HttpContext context)
{
    const int MAXFILES = 30;
    // クリーンアップします
    string path = context.Request.MapPath("Pictures");
    string[] files = System.IO.Directory.GetFiles(path);
    if (files.Length >= MAXFILES)
    {
        foreach (string fileName in files)
            System.IO.File.Delete(fileName);
    }
    // アップロードされるファイルを取得します
    for (int i = 0; i < context.Request.Files.Count && i < MAXFILES; i++)
    {
        HttpPostedFile file = context.Request.Files[i];
        try
        {
            // Image オブジェクトにファイルをロードして、ファイルが実際に画像かどうかをチェックします
            System.Drawing.Image img =
System.Drawing.Image.FromStream(file.InputStream);
            // ファイルを Pictures フォルダに保存します
            string fileName = context.Request.MapPath("Pictures/" +
file.FileName);
            file.SaveAs(fileName);
        }
        catch
        {
            // 有効な画像でない場合は、要求をスキップします
            break;
        }
    }
}
```

このコードは、最初に、画像フォルダに 30 個を超える画像がある場合に、その画像フォルダをクリーンアップします。実際のアプリケーションでは、別の方法を使用することもできます。たとえば、古い画像を圧縮したり、ユーザーごとにストレージ制限を適用します。

このコードは、クリーンアップ後に、**context.Request.Files** コレクション内のファイルをループ処理し、それらのファイルが実際に画像ファイルかどうかをチェックします。ここでは、クライアント部分のコードを実装するときにも同じチェックを追加しましたが、アプリケーションの両側を保護することが重要です。最後に、各画像を Pictures フォルダに保存します。

ここまでの成果

これで、**C1Uploader** コントロールを使ってファイルをサーバーにアップロードする Silverlight アプリケーションを作成できました。次の手順では、このアプリケーションを実行して実行時の操作を確認します。

手順 4:アプリケーションの実行

Silverlight アプリケーションを作成し、**C1Uploader** クラスを実装するためのコードを追加したので、次にアプリケーションを実行します。アプリケーションの実行時の操作を確認するには、次の手順に従います。

1. メニューから**[デバッグ]**→**[デバッグ開始]**を選択して、アプリケーションを実行します。アプリケーションが起動し、ボタンとパネルが表示されます。

2. **[画像のアップロード]**ボタンをクリックします。**[開く]**ダイアログボックスが開かれます。最初、このダイアログボックスには画像のみが表示されます。

3. .png または .jpg ファイルを見つけて選択し、**[開く]**をクリックしてこのダイアログボックスを閉じます。選択した画像がパネルに表示されていることを確認します。この画像はサーバーにアップロードされ、**Pictures** フォルダに保存されます。その後、**UploadCompleted** イベントに応答して作成した Image 要素を経由してクライアントにダウンロードし直されます。

4. **[画像のアップロード]**ボタンをクリックし、**[ファイル名]**ボックスの横にあるドロップダウンボックスで、[すべてのファイル]をクリックします。すべてのファイルが表示されます。

5. 画像以外のファイル(テキストファイルなど)を選択し、**[開く]**をクリックします。画像以外のファイルをアップロードできないことを示すメッセージが表示されます。このメッセージボックスは、前の手順のコードで指定しました。

6. **[OK]**をクリックして、メッセージボックスを閉じます。

7. **[画像のアップロード]**ボタンをクリックし、**[開く]**ダイアログボックスで、複数の画像を選択し([Ctrl]キーを押しながら画像をクリック)、**[開く]**をクリックします。複数の画像を一度にパネルに追加できることを確認します。

ここまでの成果

おめでとうございます!
これで、**Uploader for Silverlight** クイックスタートは完了です。画像ファイルをサーバーにアップロードし、それらの画像をフォルダに保存し、クライアントにダウンロードし直すことができる基本的なアップロードアプリケーションを作成しました。Silverlight アプリケーションを作成し、アプリケーションにコードを追加し、サーバーコードを実装しました。その後、アプリケーションを実行して、実行時の操作を確認しました。

Uploader の使い方

Uploader for Silverlight の**C1Uploader** コンポーネントを使用すると、簡単かつ信頼性の高い方法で、ファイルやストリームをサーバーにアップロードできます。コードを使用して、**C1Uploader** に簡単にアクセスおよび実装することができます。

基本的なプロパティ

Uploader for Silverlight には、コントロールの機能を設定するためのいくつかのプロパティがあります。主要なプロパティを次に示します。

次のプロパティを使用して、**C1Uploader** コントロールをカスタマイズできます。

プロパティ	説明
Files	アップロードされるファイルのコレクションを取得します。
Headers	アップロードごとにサーバーに送信されるヘッダーを取得します。
IsBusy	コンポーネントが現在ファイルをアップロードしているかどうかを取得します。
MaximumUploadSize	1回のアップロードで許可される最大サイズ（バイト単位）を取得または設定します。詳細については、「最大アップロードサイズ」を参照してください。
Parameters	アップロードごとにサーバーに送信されるパラメータを取得します。
Settings	**C1Uploader** を設定するために使用される **UploadManagerSettings** を取得します。
UserState	**UploadCompleted** および **UploadProgressChanged** イベントに渡される任意のオブジェクトを取得または設定します。

基本的なメソッド

Uploader for Silverlight には、このコントロールでどのファイルをいつアップロードするかを決定するためのメソッドがあります。主要なメソッドを次に示します。

次のメソッドを使用して、C1Uploader コントロールをカスタマイズできます。

メソッド	説明
AddFile	**BeginUploadFiles** メソッドが呼び出されるときにサーバーに送信される **Files** コレクションに、ファイルを追加します。
AddFiles	**BeginUploadFiles** メソッドが呼び出されるときにサーバーに送信される **Files** コレクションに、ファイルのコレクションを追加します。
BeginUploadFiles	**Files** プロパティで指定されたファイルのアップロードを開始します。
Cancel	現在のアップロード処理を取り消します。

基本的なイベント

Uploader for Silverlight には、コントロールの操作を設定およびカスタマイズするためのイベントがあります。主要なイベントを次に示します。

次のイベントを使用して、**C1Uploader** コントロールをカスタマイズできます。

イベント	説明
UploadCompleted	アップロードが完了したときに発生するイベント。
UploadProgressChanged	アップロード処理が進行したときに発生するイベント。

ファイルのアップロード

Files コレクションは、どのファイルをサーバーにアップロードするかを決定します。**Files** コレクションには、1つまたは複数のファイルを簡単に追加することができます。このコレクションに1つのファイルを追加するには、**AddFile** メソッドを使用します。複数のファイルを追加する場合は、**AddFiles** メソッドを使用できます。たとえば、ページに **C1FilePicker** コントロールと1つのボタンを置くとします。このボタンをクリックすると、選択されたファイルが **AddFile** メソッドを使って **Files** コレクションに追加されます。

1つまたは複数のファイルを選択したら、**BeginUploadFiles** メソッドを使用して、サーバーへのファイルのアップロードを開始できます。**BeginUploadFiles** メソッドが呼び出されると、**C1Uploader** コントロールは、**Files** コレクションに含まれるファイルをサーバーに送信します。何らかの理由でアップロード操作をキャンセルする必要がある場合は、**Cancel** メソッドを呼び出すことができます。たとえば、操作に時間がかかり過ぎる場合や、別のファイルのアップロード操作が既に進行中である場合などに、操作をキャンセルすることができます。

Silverlight とローカルファイル

Silverlight では、ユーザーのコンピュータのローカルファイルにアクセスする方法が制限されます。セキュリティ上の理由により、Silverlight は、ローカルファイルに直接アクセスしたり、ユーザーのファイルシステムに関する情報にアクセスすることはできません。このため、**C1Uploader** クラスは、SilverlightOpenFileDialog クラスを使用して、ファイルをアップロードするためのファイルストリームにアクセスします。

OpenFileDialog クラスは、選択されたファイルへのストリームを返します。ユーザーのファイルシステムに関する情報を提供することはありません。**OpenFileDialog** クラスは、ユーザーがローカルコンピュータまたはネットワークに接続されているコンピュータ上にある1つまたは複数のファイルを選択できるようにします。ユーザーには、選択したファイルへのアクセス権を付与するためのダイアログボックスが表示されます。これらのファイルは、分離ストレージと同様に、ファイルストリームとしてのみ公開されます。この **OpenFileDialog** クラスと **C1Uploader** クラスを使用すると、ローカルファイルをサーバーにアップロードできます。

最大アップロードサイズ

MaximumUploadSize プロパティを使用して、1回のアップロード要求でサーバーにアップロードできる最大ファイルサイズを設定できます。**MaximumUploadSize** プロパティは、サーバーで許可される1回の要求の最大サイズに設定する必要があります（ASP.NET を使用している場合、デフォルトは 4MB）。これは、このサイズより大きなファイルをアップロードできないということではなく、ファイルが **MaximumUploadSize** 以下のサイズに分割されるということです。

デフォルトでは、ファイルの最大アップロードサイズは指定されていないため、任意の大きさのファイルをアップロードできますが、ファイルチャンクのサイズを制限すると便利なことがあります。大きなファイルほどアップロードに時間がかかります。このため、ファイルのアップロードに時間がかかり過ぎたり、アップロードがタイムアウトになるという問題が発生する可能性があります。このような問題を解決するには、**C1Uploader** を **SplitFilesIntoMultipleRequests** モードに設定します。これで、**C1Uploader** はファイルを複数の要求に分割し、各要求のサイズは **MaximumUploadSize** プロパティによって決まります。

パラメータ

Parameters プロパティを使用すると、アップロードファイルに何らかの情報を関連付けておきたい場合に、クライアントからサーバーにカスタムパラメータを送信できます。たとえば、**C1Uploader** を使用して、ユーザーが音楽ファイルやビデオファイルをアップロードできるようにするとします。その際、データベースを作成するために、ユーザーにファイルに関する情報（音楽ファイルの場合は、アーティスト、タイトル、説明などのメタデータ）を要求できます。

たとえば、パラメータを設定するには、次のコードを使用します。

VisualBasic

```
uploader1.Parameters("Company") = "ComponentOne"
```

C#

```
uploader1.Parameters["Company"] = "ComponentOne";
```

これは、アップロードするファイルに "Company" パラメータを追加します。

タスク別ヘルプ

次のタスク別ヘルプトピックは、ユーザーの皆様が Visual Studio および Expression Blend に精通しており、C1Uploader 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。**Uploader for Silverlight** 製品を初めて使用される場合は、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、**Uploader 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、タスク別ヘルプトピックは、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。

ファイルを分割してアップロードする

大きなファイルほどアップロードに時間がかかります。このため、ファイルのアップロードに時間がかかり過ぎたり、アップロードがタイムアウトになるという問題が発生する可能性があります。このような問題を解決するには、**C1Uploader** を**SplitFilesIntoMultipleRequests** モードに設定します。これで、**C1Uploader** はファイルを複数の要求に分割し、各要求のサイズは **MaximumUploadSize** プロパティによって決まります。

たとえば、アップロードするファイルを複数の要求に分割する新しい **C1Uploader** コントロールを作成するには、次のコードを使用します。

VisualBasic

```
Dim uploader1 As New C1UploaderPost(FilesPerRequest.SplitFilesIntoMultipleRequests)
```

C#

```
C1UploaderPost uploader1 = new
C1UploaderPost(FilesPerRequest.SplitFilesIntoMultipleRequests);
```

C1Uploader は、アップロードするファイルを複数の要求に分割します。**MaximumUploadSize** プロパティを設定すると、1 回の要求のサイズを指定できます。

ファイルアップロードをキャンセルする

ファイルのアップロードをキャンセルする場合があります。**Cancel** メソッドを使用すると、ファイルアップロードをキャンセルできます。たとえば、アップロードに時間がかかり過ぎる場合や、多くのリソースが使用されてしまう場合に、アップロードをキャンセルできます。

たとえば、アップロードをキャンセルするには、次のコードを使用します。

VisualBasic

```
uploader1.Cancel()
```

C#

```
uploader1.Cancel();
```

これで、**C1Uploader** コントロールの現在のアップロードプロセスがキャンセルされます。

ビジー状態を確認する

アプリケーション内で、**C1Uploader** コントロールが現在ビジー状態かどうかを確認することができます。たとえば、ファイルをアップロード中かどうか、**C1Uploader** がファイルのアップロードを完了したかどうかなどを確認できます。それには、**IsBusy** プロパティを使用します。次のコードは、**C1Uploader** コントロールが現在ビジー状態かどうかを確認し、そうでない場合は新しいファイルをアップロードします。

アップローダーがビジー状態かどうかを確認するには、次の手順に従います。

VisualBasic

```
If uploader.IsBusy = False Then
    uploader.AddFile("test.pdf", stream)
    uploader.BeginUploadFiles()
End If
```

C#

```
if (uploader.IsBusy == false)
{
    uploader.AddFile("test.pdf", stream);
    uploader.BeginUploadFiles();
}
```

ファイルを圧縮してアップロードする

ファイルを圧縮してからサーバーにアップロードすることができます。それには、**ComponentOne Studio for Silverlight** に含まれる **C1Zip** クラスと**C1Uploader** クラスを使用します。このトピックでは、ファイルを圧縮してアップロードするアプリケーションを作成します。

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の左ペインから言語を選択し、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。
3. **[OK]**をクリックしてデフォルト設定を受け入れ、**[新しい Silverlight アプリケーション]**ダイアログボックスを閉じると、プロジェクトが作成されます。**MainPage.xaml** ファイルが開きます。
4. ソリューションエクスプローラウィンドウでプロジェクトを右クリックし、**[参照の追加]**を選択します。
5. **[参照の追加]**ダイアログボックスで、**C1.Silverlight.Zip.dll** および **C1.Silverlight.Uploader.dll** アセンブリを見つけて選し、**[OK]**をクリックしてプロジェクトに参照を追加します。
6. プロジェクトの XAML ウィンドウで、<Grid> タグと </Grid> タグを削除します。これらのタグは、次の手順で書き換えます。
7. 次の XAML マークアップを **MainPage.xaml** ファイルの <UserControl> タグと </UserControl> タグの間に追加しま

す。

XAML

```
   <StackPanel Orientation="Horizontal" x:Name="LayoutRoot" Background="White"
HorizontalAlignment="Center" VerticalAlignment="Center" MinHeight="400">
    <Border Padding="5" BorderBrush="Black" BorderThickness="1" Margin="2"
MinWidth="200">
        <StackPanel>
            <Button Content="ファイルを開く" Click="OpenFiles" />
            <TextBlock Text="開いているファイル:" FontWeight="Bold" Margin="10"/>
            <ItemsControl Name="files"/>
        </StackPanel>
    </Border>
    <Button Content="圧縮してファイルをアップロードする >" Grid.Column="1"
VerticalAlignment="Center" Margin="10" Click="ZipAndUpload"
Name="zipAndUploadButton"/>
    <Border Padding="5" BorderBrush="Black" BorderThickness="1" Margin="2"
Grid.Column="2">
        <StackPanel>
            <TextBlock Text="サーバー上の圧縮されたファイル:" FontWeight="Bold"
Margin="10"/>
            <ItemsControl Name="zips" Background="White"/>
        </StackPanel>
    </Border>
</StackPanel>
```

このマークアップは、いくつかのパネルと1つのボタンを作成します。ユーザーがこのボタンをクリックすると、ファイルが圧縮されてサーバーにアップロードされます。

8. ソリューションエクスプローラで、**MyProject.Web** アプリケーションを右クリックします。"MyProject" には、実際のプロジェクトアプリケーションの名前が入ります。**[新しいフォルダ]**を選択し、フォルダ名を "Zips" に指定します。このフォルダには、圧縮されたファイルが置かれます。
9. ソリューションエクスプローラで、**MainPage.xaml** ファイルを右クリックして**[コードの表示]**を選択し、コードビューに切り替えます。
10. コードビューで、次の Imports または using 文をページの先頭に追加します（ページに含まれていない場合）。

VisualBasic

```
Imports System.IO
Imports C1.C1Zip
Imports C1.Silverlight.Uploader
```

C#

```
using System.IO;
using C1.C1Zip;
using C1.Silverlight.Uploader;
```

11. **MainPage** クラスのすぐ内側で、次の変数を定義します。

VisualBasic

```
Dim fileInfos As IEnumerable(Of FileInfo)
Dim zipCount As Integer = 0
```

C#

```
IEnumerable fileInfos;
int zipCount = 0;
```

12. **MainPage.xaml.cs** または MainPage.xaml.vb ファイルで **MainPage** クラスの他のすべてのメソッドの下に、次のイベントハンドラを追加します。

VisualBasic

```
Private Sub OpenFiles(ByVal sender As Object, ByVal e As RoutedEventArgs)
    Dim dialog = New OpenFileDialog()
    dialog.ShowDialog()
    If dialog.Files Is Nothing Then
        Exit Sub
    End If
    fileInfos = dialog.Files
    files.ItemsSource = fileInfos.[Select](Function(fi) fi.Name)
    zipAndUploadButton.IsEnabled = fileInfos.Any()
End Sub
```

C#

```
private void OpenFiles(object sender, RoutedEventArgs e)
{
    var dialog = new OpenFileDialog { Multiselect = true };
    dialog.ShowDialog();
    if (dialog.Files == null)
    {
        return;
    }
    fileInfos = dialog.Files;
    files.ItemsSource = fileInfos.Select(fi => fi.Name);
    zipAndUploadButton.IsEnabled = fileInfos.Any();
}
```

このイベントハンドラは、最初に、ユーザーがアップロードするファイルを選択するための **OpenFileDialog** を表示します。

13. アップロードファイルを圧縮する次のイベントハンドラをプロジェクトに追加します。

VisualBasic

```
Private Sub ZipAndUpload(ByVal sender As Object, ByVal e As RoutedEventArgs)
    ' メモリストリームと zip ファイルを作成します
    Dim stream = New MemoryStream()
    Dim zip = New C1ZipFile(stream, True)
    ' 圧縮するファイルを追加します
    For Each fileInfo In fileInfos
        Dim fileStream = fileInfo.OpenRead()
        zip.Entries.Add(fileStream, fileInfo.Name)
        fileStream.Close()
    Next
    ' アップローダーを作成します
    Dim uploader As C1Uploader = New
C1UploaderPost(BuildAbsoluteUri("Upload.ashx"))
    ' ストリームを先頭に設定し、アップローダーにストリームを追加します
    stream.Seek(0, SeekOrigin.Begin)
    Dim zipName = System.Threading.Interlocked.Increment(zipCount) & ".zip"
    uploader.AddFile(zipName, stream)
    ' アップロードが完了したら、Zips リストに表示します
        Dim hb As New HyperlinkButton
        hb.Content = zipName
        hb.NavigateUri = BuildAbsoluteUri("Zips/" + zipName)
        If uploader.IsBusy = False Then
            Me.zips.Items.Add(hb)
        End If
    ' zip ファイルをアップロードします
    uploader.BeginUploadFiles()
End Sub
```

C#

```
private void ZipAndUpload(object sender, RoutedEventArgs e)
{
    // メモリストリームと zip ファイルを作成します
    var stream = new MemoryStream();
    var zip = new C1ZipFile(stream, true);
    // 圧縮するファイルを追加します
    foreach (var fileInfo in fileInfos)
    {
        var fileStream = fileInfo.OpenRead();
        zip.Entries.Add(fileStream, fileInfo.Name);
        fileStream.Close();
    }
    // アップローダーを作成します
    C1Uploader uploader = new C1UploaderPost(BuildAbsoluteUri("Upload.ashx"));
    // ストリームを先頭に設定し、アップローダーにストリームを追加します
    stream.Seek(0, SeekOrigin.Begin);
    var zipName = ++zipCount + ".zip";
    uploader.AddFile(zipName, stream);
```

```
    // アップロードが完了したら、Zips リストに表示します
    uploader.UploadCompleted += (s, e2) =>
    {
        zips.Items.Add(new HyperlinkButton
        {
            Content = zipName,
            NavigateUri = BuildAbsoluteUri("Zips/" + zipName)
        });
    };
    // zip ファイルをアップロードします
    uploader.BeginUploadFiles();
}
```

14. プロジェクトに次のイベントハンドラを追加します。

VisualBasic

```
Public Shared Function BuildAbsoluteUri(ByVal relativeUri As String) As Uri
    ' 現在の絶対 URI を取得します。これは、アプリケーションが展開される場所に依存します
    Dim uri As Uri = System.Windows.Browser.HtmlPage.Document.DocumentUri
    Dim uriString As String = uri.AbsoluteUri
    ' ページ名を相対サービス URI に置き換えます
    Dim ls As Integer = uriString.LastIndexOf("/"c)
    uriString = uriString.Substring(0, ls + 1) + relativeUri
    ' 新しい URI を返します
    Return New Uri(uriString, UriKind.Absolute)
End Function
```

C#

```
public static Uri BuildAbsoluteUri(string relativeUri)
{
    // 現在の絶対 URI を取得します。これは、アプリケーションが展開される場所に依存します
    Uri uri = System.Windows.Browser.HtmlPage.Document.DocumentUri;
    string uriString = uri.AbsoluteUri;
    // ページ名を相対サービス URI に置き換えます
    int ls = uriString.LastIndexOf('/');
    uriString = uriString.Substring(0, ls + 1) + relativeUri;
    // 新しい URI を返します
    return new Uri(uriString, UriKind.Absolute);
}
```

15. Visual Studio のソリューションエクスプローラで、**MyProject.Web** ソリューションを右クリックします。"MyProject" には、実際のプロジェクト名が入ります。**[追加]**→**[新しい項目]**オプションを選択します。

16. **[新しい項目の追加]**ダイアログボックスで、**[ジェネリックハンドラー]**テンプレートを選択し、新しいクラス名を "Upload.ashx" に指定します。これは、前に **C1Uploader** オブジェクトを作成した際に使用した URI です。

17. **[追加]**をクリックしてプロジェクトにファイルを追加したら、**[新しい項目の追加]**ダイアログボックスを閉じます。

18. **Upload.ashx** コードファイルを開き、次の Imports または using 文が含まれていない場合は、これをページの先頭に

追加します。

VisualBasic

```
Imports System.Web.Services
```

C#

```
using System.Web.Services;
```

19. **ProcessRequest** メソッドのデフォルトの実装を次のコードに置き換えます。

VisualBasic

```
Sub ProcessRequest(ByVal context As HttpContext) Implements
IHttpHandler.ProcessRequest
    Const MAXFILES As Integer = 30
    ' クリーンアップします
    Dim path As String = context.Request.MapPath("Zips")
    Dim files As String() = System.IO.Directory.GetFiles(path)
    If files.Length >= MAXFILES Then
        For Each fileName As String In files
        System.IO.File.Delete(fileName)
        Next
    End If
    ' アップロードされるファイルを取得します
    Dim i As Integer = 0
    While i < context.Request.Files.Count AndAlso i < MAXFILES
        Dim file As HttpPostedFile = context.Request.Files(i)
        ' ファイルを Zips フォルダに保存します
        Dim fileName As String = context.Request.MapPath("Zips/" &
file.FileName)
        file.SaveAs(fileName)
        i += 1
    End While
End Sub
```

C#

```
public void ProcessRequest(HttpContext context)
{
    const int MAXFILES = 30;
    // クリーンアップします
    string path = context.Request.MapPath("Zips");
    string[] files = System.IO.Directory.GetFiles(path);
    if (files.Length >= MAXFILES)
    {
```

```
            foreach (string fileName in files)
                System.IO.File.Delete(fileName);
        }
        // アップロードされるファイルを取得します
        for (int i = 0; i < context.Request.Files.Count && i < MAXFILES; i++)
        {
            HttpPostedFile file = context.Request.Files[i];
            // ファイルを Zips フォルダに保存します
            string fileName = context.Request.MapPath("Zips/" + file.FileName);
                file.SaveAs(fileName);
            }
        }
}
```

このコードは、最初に、画像フォルダに 30 個を超えるファイルがある場合に、その Zips フォルダをクリーンアップします。このコードは、クリーンアップ後に、**context.Request.Files** コレクション内のファイルをループ処理し、各ファイルを Zips フォルダに保存します。

ここまでの成果

これで、圧縮したファイルをアップロードする Silverlight アプリケーションを作成できました。アプリケーションの機能を確認するには、次の手順に従います。

1. **[デバッグ]**→**[デバッグ開始]**を選択して、アプリケーションを実行します。
2. **[ファイルを開く]**ボタンをクリックし、アップロードするファイルを選択します。
3. **[開く]**ダイアログボックスで、アップロードするファイルを1つ以上選択し、**[開く]**ボタンをクリックします。ダイアログボックスが閉じます。
4. **[圧縮してファイルをアップロードする]**ボタンをクリックして、選択したファイルをアップロードします。
5. 圧縮されたファイルのリンクが表示されます。最初にアップロードされたファイルは、"1.zip" になります。
6. その zip ファイルのリンクをクリックし、コンピュータに保存します。
7. その zip ファイルを展開し、アップロードした項目がファイルに含まれていることを確認します。

ファイルのアップロードとプレビュー

Uploader for Silverlight と **PDF for Silverlight** を一緒に使用することで、PDF ファイルを作成してサーバーにアップロードできます。このトピックでは、ユーザーが印刷するファイルをプレビューできるように、**C1Uploader** クラスと **C1Pdf** を使用して、ストリームにアップロードされる PDF ファイルを作成します。簡略化のために、このプロジェクトでは、**Uploader for Silverlight** サンプルに含まれる **C1UploaderHelper.cs** コードファイルを使用します。

次の手順に従います。

1. Visual Studio 2008 で、**[ファイル]**→**[新規作成]**→**[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の左ペインから言語を選択し、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。
3. **[OK]**をクリックしてデフォルト設定を受け入れ、**[新しい Silverlight アプリケーション]**ダイアログボックスを閉じると、プロジェクトが作成されます。**MainPage.xaml** ファイルが開きます。
4. ソリューションエクスプローラウィンドウでプロジェクトを右クリックし、**[参照の追加]**を選択します。

5. **[参照の追加]**ダイアログボックスで、**C1.Silverlight.Pdf.dll** および **C1.Silverlight.Uploader.dll** アセンブリを見つけて選択し、**[OK]**をクリックしてプロジェクトに参照を追加します。

6. 次の XAML マークアップを **MainPage.xaml** ファイルの <Grid> タグと </Grid> タグの間に追加します。

```
XAML
<StackPanel Width="300" HorizontalAlignment="Center" VerticalAlignment="Center">
    <TextBlock Text="Enter text here:" Margin="10"/>
    <TextBox Name="tb" Margin="10" Height="100" TextWrapping="Wrap"/>
    <Button Content="Print and Open PDF" Margin="10" Click="Button_Click" />
</StackPanel>
```

このマークアップは、PDF に入れるテキストを入力するテキストボックスと、テキストを PDF に変換してファイルをサーバーにアップロードするためのボタンを作成します。

7. ソリューションエクスプローラで、**MyProjectWeb** アプリケーションを右クリックします。"MyProject" には、実際のプロジェクトアプリケーションの名前が入ります。**[新しいフォルダ]**を選択し、フォルダ名を "temp" に指定します。このフォルダには、PDF ファイルが置かれます。

8. ソリューションエクスプローラで、**MainPage.xaml** ファイルを右クリックして[コードの表示]を選択し、コードビューに切り替えます。

9. コードビューで、次の import 文をページの先頭に追加します（ページに含まれていない場合）。

VisualBasic

```
Imports System.IO
Imports C1.Silverlight.Pdf
Imports C1.Silverlight.Uploader
```

C#

```
using System.IO;
using C1.Silverlight.Pdf;
using C1.Silverlight.Uploader;
```

10. **MainPage** クラスに次のコードを追加します。

VisualBasic

```
Private Sub Button_Click(ByVal sender As Object, ByVal e As RoutedEventArgs)
    ' PDF を作成します
    Dim doc As New C1PdfDocument()
    Dim st As String = Nothing
    st = tb.Text
    doc.DrawString(st, New Font("Arial", 14), Colors.Black, New Rect(0, 0, 500,
100))
    ' ストリームを生成します
    Dim stream As Stream
    stream = New MemoryStream()
```

```
    doc.Save(stream)
    ' ストリームをアップロードします
    Dim uploader = CreateUploader()
    uploader.AddFile("test.pdf", stream)
    uploader.BeginUploadFiles()
    AddHandler uploader.UploadCompleted, AddressOf uploader_UploadCompleted
End Sub
Private Sub uploader_UploadCompleted(ByVal sender As Object, ByVal e As
UploadCompletedEventArgs)
    ' PDF に移動します
    If e.[Error] Is Nothing Then
        Dim url = New Uri(DirectCast(e.Response, String())(0))
        System.Windows.Browser.HtmlPage.Window.Navigate(url)
    End If
End Sub
Private Function CreateUploader() As C1Uploader
    Dim mpUploader As New C1UploaderPost(FilesPerRequest.AllFilesInOneRequest)
    mpUploader.Settings.Address =
C1.Silverlight.Extensions.GetAbsoluteUri("Handler.ashx")
    Return mpUploader
End Function
```

C#

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    // PDF を作成します
    C1PdfDocument doc = new C1PdfDocument();
    doc.DrawString(tb.Text, new Font("Arial", 12), Colors.Black, new Rect(0, 0,
500, 100));
    // ストリームを生成します
    Stream stream;
    stream = new MemoryStream();
    doc.Save(stream);
    // ストリームをアップロードします
    var uploader = CreateUploader();
    uploader.AddFile("test.pdf", stream);
    uploader.BeginUploadFiles();
    uploader.UploadCompleted += new EventHandler(uploader_UploadCompleted);
}
void uploader_UploadCompleted(object sender, UploadCompletedEventArgs e)
{
    // PDF に移動します
    if (e.Error == null)
    {
        var url = new Uri(((string[])e.Response)[0]);
        System.Windows.Browser.HtmlPage.Window.Navigate(url);
    }
}
private C1Uploader CreateUploader()
{
```

```
    C1UploaderPost mpUploader = new
C1UploaderPost(FilesPerRequest.AllFilesInOneRequest);
    mpUploader.Settings.Address =
C1.Silverlight.Extensions.GetAbsoluteUri("Handler.ashx");
    return mpUploader;
}
```

11. Visual Studio のソリューションエクスプローラで、**MyProjectWeb** ソリューションを右クリックします。"MyProject" には、実際のプロジェクト名が入ります。**[追加]**→**[既存の項目]**オプションを選択します。
12. **[既存項目の追加]**ダイアログボックスで、サンプルがインストールされている ControlExplorerWeb フォルダに移動します。C1UploaderHelper.cs ファイルを選択し、**[追加]**ボタンをクリックします。デフォルトでは、**ControlExplorerWeb** フォルダは、**ComponentOne Samples\Silverlight\ControlExplorer\ControlExplorerWeb** の **Documents** フォルダに置かれています。
13. Visual Studio のソリューションエクスプローラで、**MyProjectWeb** ソリューションを右クリックします。"MyProject" には、実際のプロジェクト名が入ります。**[追加]**→**[新しい項目]**オプションを選択します。
14. **[新しい項目の追加]**ダイアログボックスで、**[ジェネリックハンドラ]**テンプレートを選択し、新しいクラスの名前を "Handler.ashx" に指定します。
15. **[追加]**をクリックしてプロジェクトにファイルを追加したら、**[新しい項目の追加]**ダイアログボックスを閉じます。
16. **Handler.ashx** コードファイルを開き、次の import 文が含まれていない場合は、これをページの先頭に追加します。"MyProject" には、ソリューションエクスプローラでの実際のプロジェクトアプリケーション名が入ります。

VisualBasic

```
Imports System.Web
Imports System.Web.Services
Imports MyProject
```

C#

```
using System.Web;
using System.Web.Services;
using MyProject;
```

17. **ProcessRequest** メソッドのデフォルトの実装を次のコードに置き換えます。

VisualBasic

```
Sub ProcessRequest(ByVal context As HttpContext) Implements
IHttpHandler.ProcessRequest
    Try
        ' カスタムパラメータを取得します
        Dim parameters As String = context.Request.Params("parameters")
        For i As Integer = 0 To context.Request.Files.Count - 1
        ' アップロードされるファイルを取得し、完全パスを計算してサーバーに保存します
        Dim file As HttpPostedFile = context.Request.Files(i)
        Dim serverFileName As String =
```

```
C1UploaderHelper.GetServerPath(context.Server, file.FileName)
        If C1UploaderHelper.ProcessPart(context, file.InputStream,
serverFileName, 1, 1) Then
            ' アップロードされるファイルの URL を応答に追加します
            Dim url As String = C1UploaderHelper.GetUploadedFileUrl(context,
"Handler.ashx", file.FileName)
            context.Response.Write((If(i > 0, "
        Else
            C1UploaderHelper.WriteError(context, C1UploaderHelper.ERROR_MESSAGE)
            Exit For
        End If
        Next
        Catch exc As Exception
        C1UploaderHelper.WriteError(context, exc.Message)
        context.Response.[End]()
    End Try
End Sub
```

C#

```
public void ProcessRequest(HttpContext context)
{
    try
    {
        // カスタムパラメータを取得します
        string parameters = context.Request.Params["parameters"];
        for (int i = 0; i < context.Request.Files.Count; i++)
        {
            // アップロードされるファイルを取得し、完全パスを計算してサーバーに保存します
            HttpPostedFile file = context.Request.Files[i];
            string serverFileName =
C1UploaderHelper.GetServerPath(context.Server, file.FileName);
            if (File.Exists(file.FileName))
                File.Delete(file.FileName);
            if (C1UploaderHelper.ProcessPart(context, file.InputStream,
serverFileName, 1, 1))
            {
                // アップロードされるファイルの URL を応答に追加します
                string url = C1UploaderHelper.GetUploadedFileUrl(context,
"Handler.ashx", file.FileName);
                context.Response.Write((i > 0 ? "
            }
            else
            {
                C1UploaderHelper.WriteError(context,
C1UploaderHelper.ERROR_MESSAGE);
                break;
            }
        }
    }
    catch (Exception exc)
```

```
    {
        C1UploaderHelper.WriteError(context, exc.Message);
        context.Response.End();
    }
}
```

ここまでの成果

これで、テキストを PDF ファイルとしてアップロードしてプレビューする Silverlight アプリケーションを作成できました。アプリケーションの機能を確認するには、次の手順に従います。

1. **[デバッグ]**→**[デバッグ開始]**を選択して、アプリケーションを実行します。
2. テキストボックスにテキスト("Hello World!" など)を入力します。
3. **[Print and Open PDF]**ボタンをクリックします。
   入力したテキストが **test.pdf** ファイルに追加されて開かれます。この PDF ファイルを閉じ、テキストボックスに別のテキストを入力し、同様にボタンをクリックします。PDF ファイルが開かれて、新しいテキストが表示されます。