PDF Combiner

A simple Windows Forms app to combine multiple
PDF files in a single PDF file

Table of Contents

1 Symbol Reference

1.1 PdfCombiner Namespace

This is namespace PdfCombiner.

Namespaces

Name	Description
Properties (⊠ see page 1)	This is namespace PdfCombiner.Properties.

Classes

	Name	Description
🔧🅐	BaseException (⊠ see page 4)	This function is used to take exceptions which can be thrown in PDF Combiner App
🔧🆂	ExceptionExtensions (⊠ see page 5)	This is class PdfCombiner.ExceptionExtensions.
🔧	FrmMain (⊠ see page 6)	This is class PdfCombiner.FrmMain.
🔧🆂	Program (⊠ see page 22)	This is class PdfCombiner.Program.

1.1.1 PdfCombiner.Properties Namespace

This is namespace PdfCombiner.Properties.

Classes

	Name	Description
🔧	Resources (⊠ see page 1)	A strongly-typed resource class, for looking up localized strings, etc.
🔧	Settings (⊠ see page 3)	This is class PdfCombiner.Properties.Settings.

1.1.1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
🔧	Resources (⊠ see page 1)	A strongly-typed resource class, for looking up localized strings, etc.
🔧	Settings (⊠ see page 3)	This is class PdfCombiner.Properties.Settings.

1.1.1.1.1 Resources Class

A strongly-typed resource class, for looking up localized strings, etc.

Class Hierarchy

PdfCombiner.Properties.Resources

C#

```
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyType
dResourceBuilder",
"15.0.0.0")]
[global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
internal class Resources;
```

File

Resources.Designer.cs (⊡ see page 37)

Description

This class was auto-generated by the StronglyTypedResourceBuilder class via a tool like ResGen or Visual Studio. To add or remove a member, edit your .ResX file then rerun ResGen with the /str option, or rebuild your VS project.

Methods

	Name	Description
⇒◆▸	Resources (⊡ see page 2)	This is Resources, a member of class Resources.

Resources Fields

	Name	Description
◆▫S	resourceCulture (⊡ see page 2)	This is resourceCulture, a member of class Resources.
◆▫S	resourceMan (⊡ see page 3)	This is resourceMan, a member of class Resources.

Resources Properties

	Name	Description
▨▸S	Culture (⊡ see page 3)	Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.
▨▸S	ResourceManager (⊡ see page 3)	Returns the cached ResourceManager instance used by this class.

1.1.1.1.1.1 Resources.Resources Constructor

C#

```
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
internal Resources();
```

Description

This is Resources, a member of class Resources.

Body Source

```
1:
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
2: internal Resources() {
3: }
```

1.1.1.1.1.2 Resources Fields

1.1.1.1.1.2.1 Resources.resourceCulture Field

C#

```
private static global::System.Globalization.CultureInfo resourceCulture;
```

Description

This is resourceCulture, a member of class Resources.

1.1.1.1.1.2.2 Resources.resourceMan Field

C#

```
private static global::System.Resources.ResourceManager resourceMan;
```

Description

This is resourceMan, a member of class Resources.

1.1.1.1.1.3 Resources Properties

1.1.1.1.1.3.1 Resources.Culture Property

Overrides the current thread's CurrentUICulture property for all resource lookups using this strongly typed resource class.

C#

```
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
internal static global::System.Globalization.CultureInfo Culture;
```

1.1.1.1.1.3.2 Resources.ResourceManager Property

Returns the cached ResourceManager instance used by this class.

C#

```
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
internal static global::System.Resources.ResourceManager ResourceManager;
```

1.1.1.1.2 Settings Class

Class Hierarchy
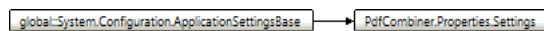
C#

```
[global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.Sett
ingsDesigner.SettingsSingleFileGenerator",
"11.0.0.0")]
internal sealed class Settings : global::System.Configuration.ApplicationSettingsBase;
```

File

Settings.Designer.cs (⊿ see page 38)

Description

This is class PdfCombiner.Properties.Settings.

Settings Fields

	Name	Description
⚙️⬛	defaultInstance (⊿ see page 3)	This is defaultInstance, a member of class Settings.

1.1.1.1.2.1 Settings Fields

1.1.1.1.2.1.1 Settings.defaultInstance Field

C#

```
private static Settings defaultInstance =
```

```
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new
Settings()))));
```

Description

This is defaultInstance, a member of class Settings.

1.1.2 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
✦🅰	BaseException (▣ see page 4)	This function is used to take exceptions which can be thrown in PDF Combiner App
✦Ⓢ	ExceptionExtensions (▣ see page 5)	This is class PdfCombiner.ExceptionExtensions.
✦	FrmMain (▣ see page 6)	This is class PdfCombiner.FrmMain.
✦Ⓢ	Program (▣ see page 22)	This is class PdfCombiner.Program.

1.1.2.1 BaseException Class

This function is used to take exceptions which can be thrown in PDF Combiner App

Class Hierarchy

Exception → PdfCombiner.BaseException

C#

```
[Serializable]
public abstract class BaseException : Exception;
```

File

BaseException.cs (▣ see page 24)

Methods

	Name	Description
⇒◆	BaseException (▣ see page 4)	This is BaseException, a member of class BaseException.

1.1.2.1.1 BaseException Constructor

1.1.2.1.1.1 BaseException.BaseException Constructor ()

C#

```
protected BaseException();
```

Description

This is BaseException, a member of class BaseException.

Body Source

```
1: protected BaseException()
2: {
3: }
```

1.1.2.1.1.2 BaseException.BaseException Constructor (string)

C#

```
protected BaseException(string message);
```

Description

This is BaseException, a member of class BaseException.

Body Source

```
1: protected BaseException(string message) : base(message)
2: {
3: }
```

1.1.2.1.1.3 BaseException.BaseException Constructor (string, Exception)

C#

```
protected BaseException(string message, Exception innerException);
```

Description

This is BaseException, a member of class BaseException.

Body Source

```
1: protected BaseException(string message, Exception innerException) : base(message,
innerException)
2: {
3: }
```

1.1.2.2 ExceptionExtensions Class

Class Hierarchy

PdfCombiner.ExceptionExtensions

C#

```
public static class ExceptionExtensions;
```

File

ExceptionExtensions.cs (☐ see page 25)

Description

This is class PdfCombiner.ExceptionExtensions.

ExceptionExtensions Methods

	Name	Description
⬧S	GetAllMessages (☐ see page 5)	This method is used to take all messages with stack trace info in exception

1.1.2.2.1 ExceptionExtensions Methods

1.1.2.2.1.1 ExceptionExtensions.GetAllMessages Method

This method is used to take all messages with stack trace info in exception

C#

```
public static string GetAllMessages(this Exception exception);
```

Parameters

Parameters	Description
this Exception exception	Exception

Returns

Detailed Error Message

Body Source

```
 1: public static string GetAllMessages(this Exception exception)
 2: {
 3:     var messages = exception.Message;
 4:     if (exception.InnerException != null)
 5:         messages += " " + exception.InnerException.GetAllMessages();
 6:     if (!string.IsNullOrEmpty(exception.StackTrace))
 7:         messages += " " + exception.StackTrace;
 8:     if (!string.IsNullOrEmpty(exception.Source))
 9:         messages += " " + exception.Source;
10:     return messages;
11: }
```

1.1.2.3 FrmMain Class

Class Hierarchy

Form ➝ PdfCombiner.FrmMain
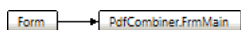
C#

```
public class FrmMain : Form;
```

File

FrmMain.Designer.cs (⊡ see page 33)

Description

This is class PdfCombiner.FrmMain.

Methods

	Name	Description
⇒◆	FrmMain (⊡ see page 7)	This is FrmMain, a member of class FrmMain.

FrmMain Fields

	Name	Description
◆	AppTitle (⊡ see page 8)	This is AppTitle, a member of class FrmMain.
◆⌂	btnAddFile (⊡ see page 8)	This is btnAddFile, a member of class FrmMain.
◆⌂	btnAddFolder (⊡ see page 8)	This is btnAddFolder, a member of class FrmMain.
◆⌂	btnClearList (⊡ see page 8)	This is btnClearList, a member of class FrmMain.
◆⌂	btnCombineITextSharp (⊡ see page 8)	This is btnCombineITextSharp, a member of class FrmMain.
◆⌂	btnCombinePdfSharp (⊡ see page 9)	This is btnCombinePdfSharp, a member of class FrmMain.
◆⌂	components (⊡ see page 9)	Required designer variable.
◆⌂	lbFiles (⊡ see page 9)	This is lbFiles, a member of class FrmMain.
◆⌂	lblDetails (⊡ see page 9)	This is lblDetails, a member of class FrmMain.
◆⌂	menuItemDelete (⊡ see page 9)	This is menuItemDelete, a member of class FrmMain.

	menuItemOrderAscending (⊡ see page 9)	This is menuItemOrderAscending, a member of class FrmMain.
	menuItemOrderDescending (⊡ see page 9)	This is menuItemOrderDescending, a member of class FrmMain.
	menuStrip (⊡ see page 10)	This is menuStrip, a member of class FrmMain.
	pbFiles (⊡ see page 10)	This is pbFiles, a member of class FrmMain.

FrmMain Methods

	Name	Description
	btnAddFile_Click (⊡ see page 10)	It is used to add single or multiple PDF files to combine When you choose file or files The listbox in the form will be filled with the name of the files Which you choose to combine
	btnAddFolder_Click (⊡ see page 10)	This function is used to add PDF files in a folder which you choose in Folder Dialog recursively When you use it the added files will be seen in listbox
	btnClearList_Click (⊡ see page 11)	This function is used to clear the file list in listbox
	btnCombineITextSharp_Click (⊡ see page 11)	This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses iTextSharp Nuget Package
	btnCombinePdfSharp_Click (⊡ see page 13)	This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses PdfSharp Nuget Package
	DeleteFilesFromListBox (⊡ see page 14)	This method is used to Delete selected files from ListBox This is used different ways in this app So we created this as a method
	Dispose (⊡ see page 15)	Clean up any resources being used.
	FrmMain_FormClosed (⊡ see page 15)	This function is used to terminate application
	FrmMain_Shown (⊡ see page 15)	This function is used to assign menu to listbox When form is shown
	InitializeComponent (⊡ see page 16)	Required method for Designer support - do not modify the contents of this method with the code editor.
	InitializeFileDialog (⊡ see page 19)	This function is used to set options of file dialog like Filter, start path etc.
	lbFiles_DragDrop (⊡ see page 19)	This is used to drag and drop an item in listbox Which is used to order books and combine them with that order
	lbFiles_DragOver (⊡ see page 19)	This is used to give effect while dragging and dropping an item in listbox
	lbFiles_KeyDown (⊡ see page 20)	This function is used to delete files in listbox which are chosen If you press the Delete button in your keyboard The selected files will be deleted from the list
	lbFiles_MouseDown (⊡ see page 20)	This is used to control mouse actions and if that is not right click Start to drag and drop item in listbox
	menuItemDelete_Click (⊡ see page 20)	This is used to delete selected items in listbox When menu item Delete is clicked and opened dialog
	menuItemOrderAscending_Click (⊡ see page 21)	This method is used to order elements in listbox ascending
	menuItemOrderDescending_Click (⊡ see page 21)	This method is used to order elements in listbox descending by name
	SortItems (⊡ see page 22)	This method is used to order items in listbox And return them with the given order as parametre

1.1.2.3.1 FrmMain.FrmMain Constructor

C#

```csharp
public FrmMain();
```

Description

This is FrmMain, a member of class FrmMain.

Body Source

```
1: public FrmMain()
2: {
3:     InitializeComponent();
4: }
```

1.1.2.3.2 FrmMain Fields

1.1.2.3.2.1 FrmMain.AppTitle Field

C#

```
public const string AppTitle = "PDF Combiner";
```

Description

This is AppTitle, a member of class FrmMain.

1.1.2.3.2.2 FrmMain.btnAddFile Field

C#

```
private System.Windows.Forms.Button btnAddFile;
```

Description

This is btnAddFile, a member of class FrmMain.

1.1.2.3.2.3 FrmMain.btnAddFolder Field

C#

```
private System.Windows.Forms.Button btnAddFolder;
```

Description

This is btnAddFolder, a member of class FrmMain.

1.1.2.3.2.4 FrmMain.btnClearList Field

C#

```
private System.Windows.Forms.Button btnClearList;
```

Description

This is btnClearList, a member of class FrmMain.

1.1.2.3.2.5 FrmMain.btnCombineITextSharp Field

C#

```
private System.Windows.Forms.Button btnCombineITextSharp;
```

Description

This is btnCombineITextSharp, a member of class FrmMain.

1.1.2.3.2.6 **FrmMain.btnCombinePdfSharp Field**

C#

```
private System.Windows.Forms.Button btnCombinePdfSharp;
```

Description

This is btnCombinePdfSharp, a member of class FrmMain.

1.1.2.3.2.7 **FrmMain.components Field**

Required designer variable.

C#

```
private System.ComponentModel.IContainer components;
```

1.1.2.3.2.8 **FrmMain.lbFiles Field**

C#

```
private System.Windows.Forms.ListBox lbFiles;
```

Description

This is lbFiles, a member of class FrmMain.

1.1.2.3.2.9 **FrmMain.lblDetails Field**

C#

```
private System.Windows.Forms.Label lblDetails;
```

Description

This is lblDetails, a member of class FrmMain.

1.1.2.3.2.10 **FrmMain.menuItemDelete Field**

C#

```
private System.Windows.Forms.ToolStripMenuItem menuItemDelete;
```

Description

This is menuItemDelete, a member of class FrmMain.

1.1.2.3.2.11 **FrmMain.menuItemOrderAscending Field**

C#

```
private System.Windows.Forms.ToolStripMenuItem menuItemOrderAscending;
```

Description

This is menuItemOrderAscending, a member of class FrmMain.

1.1.2.3.2.12 **FrmMain.menuItemOrderDescending Field**

C#

```
private System.Windows.Forms.ToolStripMenuItem menuItemOrderDescending;
```

Description

This is menuItemOrderDescending, a member of class FrmMain.

1.1.2.3.2.13 **FrmMain.menuStrip Field**

C#

```
private System.Windows.Forms.ContextMenuStrip menuStrip;
```

Description

This is menuStrip, a member of class FrmMain.

1.1.2.3.2.14 **FrmMain.pbFiles Field**

C#

```
private System.Windows.Forms.ProgressBar pbFiles;
```

Description

This is pbFiles, a member of class FrmMain.

1.1.2.3.3 **FrmMain Methods**

1.1.2.3.3.1 **FrmMain.btnAddFile_Click Method**

It is used to add single or multiple PDF files to combine When you choose file or files The listbox in the form will be filled with the name of the files Which you choose to combine

C#

```
private void btnAddFile_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```
 1: private void btnAddFile_Click(object sender, EventArgs e)
 2: {
 3:     using (var dialogAddFile = new OpenFileDialog())
 4:     {
 5:         InitializeFileDialog(dialogAddFile);
 6:         var result = dialogAddFile.ShowDialog();
 7:         var addedFileCount = 0;
 8:         if (result == DialogResult.OK && dialogAddFile.FileNames != null)
 9:         {
10:             foreach (var file in dialogAddFile.FileNames)
11:             {
12:                 if (!lbFiles.Items.Contains(file))
13:                 {
14:                     lbFiles.Items.Add(file);
15:                     addedFileCount++;
16:                 }
17:             }
18:             MessageBox.Show(addedFileCount + " file/s successfully added to the list",
AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
19:         }
20:     }
21: }
```

1.1.2.3.3.2 **FrmMain.btnAddFolder_Click Method**

This function is used to add PDF files in a folder which you choose in Folder Dialog recursively When you use it the added

files will be seen in listbox

C#

```
private void btnAddFolder_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```
 1: private void btnAddFolder_Click(object sender, EventArgs e)
 2: {
 3:     using (var dialogAddFolder = new FolderBrowserDialog())
 4:     {
 5:         var result = dialogAddFolder.ShowDialog();
 6:         if (result == DialogResult.OK &&
!string.IsNullOrEmpty(dialogAddFolder.SelectedPath))
 7:         {
 8:             var fileNames = Directory.GetFiles(dialogAddFolder.SelectedPath, "*.pdf",
SearchOption.AllDirectories);
 9:             var addedFileCount = 0;
10:             foreach (var file in fileNames)
11:             {
12:                 if (!lbFiles.Items.Contains(file))
13:                 {
14:                     lbFiles.Items.Add(file);
15:                     addedFileCount++;
16:                 }
17:             }
18:             MessageBox.Show(addedFileCount + " file/s successfully added to the list",
AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
19:         }
20:     }
21: }
```

1.1.2.3.3.3 FrmMain.btnClearList_Click Method

This function is used to clear the file list in listbox

C#

```
private void btnClearList_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```
1: private void btnClearList_Click(object sender, EventArgs e)
2: {
3:     var fileCount = lbFiles.Items.Count;
4:     lbFiles.Items.Clear();
5:     MessageBox.Show(fileCount + " file/s successfully deleted from the list", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Information);
6: }
```

1.1.2.3.3.4 FrmMain.btnCombineITextSharp_Click Method

This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses iTextSharp Nuget Package

C#

```
private void btnCombineITextSharp_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```
 1: private void btnCombineITextSharp_Click(object sender, EventArgs e)
 2: {
 3:     try
 4:     {
 5:         if (lbFiles.Items.Count < 2)
 6:             MessageBox.Show("There must be at least 2 PDF files to combine", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Error);
 7:         else
 8:         {
 9:             using (var dialogExport = new FolderBrowserDialog())
10:             {
11:                 var result = dialogExport.ShowDialog();
12:                 if (result == DialogResult.OK &&
!string.IsNullOrEmpty(dialogExport.SelectedPath))
13:                 {
14:                     var outputFileName = dialogExport.SelectedPath + "/" +
Guid.NewGuid() + ".pdf";
15:                     var fileCount = lbFiles.Items.Count;
16:                     var combinedFiles = 0;
17:
18:                     var outputFile = new Document();
19:                     using (FileStream outputFileStream = new FileStream(outputFileName,
FileMode.Create))
20:                     {
21:                         using (var pdfWriter = new
iTextSharp.text.pdf.PdfCopy(outputFile, outputFileStream))
22:                         {
23:                             if (pdfWriter == null)
24:                             {
25:                                 return;
26:                             }
27:                             pdfWriter.SetFullCompression();
28:                             outputFile.Open();
29:                             for (int i = 0; i < lbFiles.Items.Count; i++)
30:                             {
31:                                 try
32:                                 {
33:                                     var pdfReader = new
iTextSharp.text.pdf.PdfReader(lbFiles.Items[i].ToString());
34:                                     pdfReader.ConsolidateNamedDestinations();
35:                                     for (int j = 1; j <= pdfReader.NumberOfPages; j++)
36:                                     {
37:                                         var page = pdfWriter.GetImportedPage(pdfReader,
j);
38:                                         pdfWriter.AddPage(page);
39:                                     }
40:                                     pdfReader.Close();
41:                                     combinedFiles++;
42:                                 }
43:                                 catch (Exception)
44:                                 {
45:                                     fileCount--;
46:                                 }
47:
48:                                 pbFiles.Value = (combinedFiles * 100 / fileCount);
49:                                 if (pbFiles.Value > pbFiles.Maximum)
```

```
50:                                        pbFiles.Value = pbFiles.Maximum;
51:                                    ActiveForm.Text = "%" + pbFiles.Value;
52:                                }
53:                                pdfWriter.Close();
54:                            }
55:                            outputFile.Close();
56:                        }
57:
58:                        MessageBox.Show(fileCount + " PDF files successfully combined in "
+ outputFileName, AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
59:
60:                        pbFiles.Value = pbFiles.Minimum;
61:                        ActiveForm.Text = AppTitle;
62:                    }
63:                }
64:            }
65:        }
66:        catch (Exception ex)
67:        {
68:            MessageBox.Show("There is an error while combining PDF files in list. Details:
" + ex.GetAllMessages(), AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Error);
69:            pbFiles.Value = pbFiles.Minimum;
70:            ActiveForm.Text = AppTitle;
71:        }
72: }
```

1.1.2.3.3.5 FrmMain.btnCombinePdfSharp_Click Method

This function is used to take list of PDF files in listbox, Combine them with a single file in location which you choose in folder dialog And you can see the progress in progress bar It uses PdfSharp Nuget Package

C#

```
private void btnCombinePdfSharp_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```
 1: private void btnCombinePdfSharp_Click(object sender, EventArgs e)
 2: {
 3:     try
 4:     {
 5:         if (lbFiles.Items.Count < 2)
 6:             MessageBox.Show("There must be at least 2 PDF files to combine", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Error);
 7:         else
 8:         {
 9:             using (var dialogExport = new FolderBrowserDialog())
10:             {
11:                 var result = dialogExport.ShowDialog();
12:                 if (result == DialogResult.OK &&
!string.IsNullOrEmpty(dialogExport.SelectedPath))
13:                 {
14:                     var outputFileName = dialogExport.SelectedPath + "/" +
Guid.NewGuid() + ".pdf";
15:                     using (var outputFile = new PdfDocument())
16:                     {
17:                         outputFile.Options.CompressContentStreams = true;
18:                         outputFile.Options.EnableCcittCompressionForBilevelImages =
true;
19:                         outputFile.Options.FlateEncodeMode =
PdfFlateEncodeMode.BestCompression;
20:                         var fileCount = lbFiles.Items.Count;
```

```
21:                            var combinedFiles = 0;
22:
23:                            for (int i = 0; i < lbFiles.Items.Count; i++)
24:                            {
25:                                try
26:                                {
27:                                    var inputDocument =
PdfReader.Open(lbFiles.Items[i].ToString(), PdfDocumentOpenMode.Import);
28:                                    var count = inputDocument.PageCount;
29:                                    for (int idx = 0; idx < count; idx++)
30:                                    {
31:                                        var page = inputDocument.Pages[idx];
32:                                        outputFile.AddPage(page);
33:                                    }
34:                                    inputDocument.Close();
35:                                    combinedFiles++;
36:                                }
37:                                catch (Exception)
38:                                {
39:                                    fileCount--;
40:                                }
41:
42:                                pbFiles.Value = (combinedFiles * 100 / fileCount);
43:                                if (pbFiles.Value > pbFiles.Maximum)
44:                                    pbFiles.Value = pbFiles.Maximum;
45:                                ActiveForm.Text = "%" + pbFiles.Value;
46:                            }
47:                            outputFile.Save(outputFileName);
48:                            outputFile.Close();
49:
50:                            MessageBox.Show(fileCount + " PDF files successfully combined
in " + outputFileName, AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
51:                        }
52:
53:                        pbFiles.Value = pbFiles.Minimum;
54:                        ActiveForm.Text = AppTitle;
55:                    }
56:                }
57:            }
58:        }
59:    catch (Exception ex)
60:    {
61:        MessageBox.Show("There is an error while combining PDF files in list. Details:
" + ex.GetAllMessages(), AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Error);
62:        pbFiles.Value = pbFiles.Minimum;
63:        ActiveForm.Text = AppTitle;
64:    }
65: }
```

1.1.2.3.3.6 FrmMain.DeleteFilesFromListBox Method

This method is used to Delete selected files from ListBox This is used different ways in this app So we created this as a method

C#

```
private void DeleteFilesFromListBox();
```

Body Source

```
1: private void DeleteFilesFromListBox()
2: {
3:     var allItems = new List<string>();
4:     var removedItems = new List<string>();
5:     var fileCount = lbFiles.SelectedItems.Count;
6:     for (int i = 0; i < lbFiles.SelectedItems.Count; i++)
7:     {
8:         removedItems.Add(lbFiles.SelectedItems[i].ToString());
9:     }
```

```
10:      foreach (var item in lbFiles.Items)
11:      {
12:          if (!removedItems.Any(j => j == item.ToString()))
13:              allItems.Add(item.ToString());
14:      }
15:      lbFiles.Items.Clear();
16:      foreach (var item in allItems)
17:      {
18:          lbFiles.Items.Add(item);
19:      }
20:      MessageBox.Show(fileCount + " file/s are deleted from the list.", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Information);
21: }
```

1.1.2.3.3.7 FrmMain.Dispose Method

Clean up any resources being used.

C#

```
protected override void Dispose(bool disposing);
```

Parameters

Parameters	Description
bool disposing	true if managed resources should be disposed; otherwise, false.

Body Source

```
1: protected override void Dispose(bool disposing)
2: {
3:      if (disposing && (components != null))
4:      {
5:          components.Dispose();
6:      }
7:      base.Dispose(disposing);
8: }
```

1.1.2.3.3.8 FrmMain.FrmMain_FormClosed Method

This function is used to terminate application

C#

```
private static void FrmMain_FormClosed(object sender, FormClosedEventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
FormClosedEventArgs e	Event Arguments

Body Source

```
1: private static void FrmMain_FormClosed(object sender, FormClosedEventArgs e)
2: {
3:      MessageBox.Show("Thanks for using this app", AppTitle, MessageBoxButtons.OK,
MessageBoxIcon.Information);
4:      Application.Exit();
5: }
```

1.1.2.3.3.9 FrmMain.FrmMain_Shown Method

This function is used to assign menu to listbox When form is shown

C#

```
private void FrmMain_Shown(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```
1: private void FrmMain_Shown(object sender, EventArgs e)
2: {
3:      lbFiles.ContextMenuStrip = menuStrip;
4: }
```

1.1.2.3.3.10 FrmMain.InitializeComponent Method

Required method for Designer support - do not modify the contents of this method with the code editor.

C#

```
private void InitializeComponent();
```

Body Source

```
 1: private void InitializeComponent()
 2: {
 3:      this.components = new System.ComponentModel.Container();
 4:      System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(FrmMain));
 5:      this.btnCombinePdfSharp = new System.Windows.Forms.Button();
 6:      this.btnAddFolder = new System.Windows.Forms.Button();
 7:      this.btnAddFile = new System.Windows.Forms.Button();
 8:      this.btnClearList = new System.Windows.Forms.Button();
 9:      this.lbFiles = new System.Windows.Forms.ListBox();
10:      this.lblDetails = new System.Windows.Forms.Label();
11:      this.pbFiles = new System.Windows.Forms.ProgressBar();
12:      this.btnCombineITextSharp = new System.Windows.Forms.Button();
13:      this.menuStrip = new System.Windows.Forms.ContextMenuStrip(this.components);
14:      this.menuItemDelete = new System.Windows.Forms.ToolStripMenuItem();
15:      this.menuItemOrderAscending = new System.Windows.Forms.ToolStripMenuItem();
16:      this.menuItemOrderDescending = new System.Windows.Forms.ToolStripMenuItem();
17:      this.menuStrip.SuspendLayout();
18:      this.SuspendLayout();
19:      //
20:      // btnCombinePdfSharp
21:      //
22:      this.btnCombinePdfSharp.Image =
((System.Drawing.Image)(resources.GetObject("btnCombinePdfSharp.Image")));
23:      this.btnCombinePdfSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
24:      this.btnCombinePdfSharp.Location = new System.Drawing.Point(442, 5);
25:      this.btnCombinePdfSharp.Name = "btnCombinePdfSharp";
26:      this.btnCombinePdfSharp.Size = new System.Drawing.Size(214, 37);
27:      this.btnCombinePdfSharp.TabIndex = 0;
28:      this.btnCombinePdfSharp.Text = "Combine Files (PdfSharp)";
29:      this.btnCombinePdfSharp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
30:      this.btnCombinePdfSharp.UseVisualStyleBackColor = true;
31:      this.btnCombinePdfSharp.Click += new
System.EventHandler(this.btnCombinePdfSharp_Click);
32:      //
33:      // btnAddFolder
34:      //
35:      this.btnAddFolder.Image =
((System.Drawing.Image)(resources.GetObject("btnAddFolder.Image")));
36:      this.btnAddFolder.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
37:      this.btnAddFolder.Location = new System.Drawing.Point(12, 45);
38:      this.btnAddFolder.Name = "btnAddFolder";
39:      this.btnAddFolder.Size = new System.Drawing.Size(105, 37);
40:      this.btnAddFolder.TabIndex = 2;
41:      this.btnAddFolder.Text = "Add Folder";
```

```
 42:        this.btnAddFolder.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
 43:        this.btnAddFolder.UseVisualStyleBackColor = true;
 44:        this.btnAddFolder.Click += new System.EventHandler(this.btnAddFolder_Click);
 45:        //
 46:        // btnAddFile
 47:        //
 48:        this.btnAddFile.Image =
((System.Drawing.Image)(resources.GetObject("btnAddFile.Image")));
 49:        this.btnAddFile.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
 50:        this.btnAddFile.Location = new System.Drawing.Point(12, 5);
 51:        this.btnAddFile.Name = "btnAddFile";
 52:        this.btnAddFile.Size = new System.Drawing.Size(105, 37);
 53:        this.btnAddFile.TabIndex = 3;
 54:        this.btnAddFile.Text = "Add Files";
 55:        this.btnAddFile.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
 56:        this.btnAddFile.UseVisualStyleBackColor = true;
 57:        this.btnAddFile.Click += new System.EventHandler(this.btnAddFile_Click);
 58:        //
 59:        // btnClearList
 60:        //
 61:        this.btnClearList.Image =
((System.Drawing.Image)(resources.GetObject("btnClearList.Image")));
 62:        this.btnClearList.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
 63:        this.btnClearList.Location = new System.Drawing.Point(123, 5);
 64:        this.btnClearList.Name = "btnClearList";
 65:        this.btnClearList.Size = new System.Drawing.Size(105, 37);
 66:        this.btnClearList.TabIndex = 4;
 67:        this.btnClearList.Text = "Clear File List";
 68:        this.btnClearList.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
 69:        this.btnClearList.UseVisualStyleBackColor = true;
 70:        this.btnClearList.Click += new System.EventHandler(this.btnClearList_Click);
 71:        //
 72:        // lbFiles
 73:        //
 74:        this.lbFiles.AllowDrop = true;
 75:        this.lbFiles.FormattingEnabled = true;
 76:        this.lbFiles.Location = new System.Drawing.Point(12, 88);
 77:        this.lbFiles.Name = "lbFiles";
 78:        this.lbFiles.SelectionMode = System.Windows.Forms.SelectionMode.MultiExtended;
 79:        this.lbFiles.Size = new System.Drawing.Size(644, 186);
 80:        this.lbFiles.TabIndex = 5;
 81:        this.lbFiles.DragDrop += new
System.Windows.Forms.DragEventHandler(this.lbFiles_DragDrop);
 82:        this.lbFiles.DragOver += new
System.Windows.Forms.DragEventHandler(lbFiles_DragOver);
 83:        this.lbFiles.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.lbFiles_KeyDown);
 84:        this.lbFiles.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.lbFiles_MouseDown);
 85:        //
 86:        // lblDetails
 87:        //
 88:        this.lblDetails.AutoSize = true;
 89:        this.lblDetails.Location = new System.Drawing.Point(12, 65);
 90:        this.lblDetails.Name = "lblDetails";
 91:        this.lblDetails.Size = new System.Drawing.Size(0, 13);
 92:        this.lblDetails.TabIndex = 6;
 93:        //
 94:        // pbFiles
 95:        //
 96:        this.pbFiles.Location = new System.Drawing.Point(123, 45);
 97:        this.pbFiles.Name = "pbFiles";
 98:        this.pbFiles.Size = new System.Drawing.Size(313, 37);
 99:        this.pbFiles.TabIndex = 7;
100:        //
101:        // btnCombineITextSharp
102:        //
103:        this.btnCombineITextSharp.Image =
((System.Drawing.Image)(resources.GetObject("btnCombineITextSharp.Image")));
```

1

```
104:        this.btnCombineITextSharp.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
105:        this.btnCombineITextSharp.Location = new System.Drawing.Point(442, 45);
106:        this.btnCombineITextSharp.Name = "btnCombineITextSharp";
107:        this.btnCombineITextSharp.Size = new System.Drawing.Size(214, 37);
108:        this.btnCombineITextSharp.TabIndex = 9;
109:        this.btnCombineITextSharp.Text = "Combine Files (iTextSharp)";
110:        this.btnCombineITextSharp.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
111:        this.btnCombineITextSharp.UseVisualStyleBackColor = true;
112:        this.btnCombineITextSharp.Click += new
System.EventHandler(this.btnCombineITextSharp_Click);
113:        //
114:        // menuStrip
115:        //
116:        this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
117:        this.menuItemDelete,
118:        this.menuItemOrderAscending,
119:        this.menuItemOrderDescending});
120:        this.menuStrip.Name = "menuStrip";
121:        this.menuStrip.Size = new System.Drawing.Size(178, 70);
122:        //
123:        // menuItemDelete
124:        //
125:        this.menuItemDelete.Image =
((System.Drawing.Image)(resources.GetObject("menuItemDelete.Image")));
126:        this.menuItemDelete.Name = "menuItemDelete";
127:        this.menuItemDelete.Size = new System.Drawing.Size(177, 22);
128:        this.menuItemDelete.Text = "Delete";
129:        this.menuItemDelete.Click += new System.EventHandler(this.menuItemDelete_Click);
130:        //
131:        // menuItemOrderAscending
132:        //
133:        this.menuItemOrderAscending.Image =
((System.Drawing.Image)(resources.GetObject("menuItemOrderAscending.Image")));
134:        this.menuItemOrderAscending.Name = "menuItemOrderAscending";
135:        this.menuItemOrderAscending.Size = new System.Drawing.Size(177, 22);
136:        this.menuItemOrderAscending.Text = "Order (Ascending)";
137:        this.menuItemOrderAscending.Click += new
System.EventHandler(this.menuItemOrderAscending_Click);
138:        //
139:        // menuItemOrderDescending
140:        //
141:        this.menuItemOrderDescending.Image =
((System.Drawing.Image)(resources.GetObject("menuItemOrderDescending.Image")));
142:        this.menuItemOrderDescending.Name = "menuItemOrderDescending";
143:        this.menuItemOrderDescending.Size = new System.Drawing.Size(177, 22);
144:        this.menuItemOrderDescending.Text = "Order (Descending)";
145:        this.menuItemOrderDescending.Click += new
System.EventHandler(this.menuItemOrderDescending_Click);
146:        //
147:        // FrmMain
148:        //
149:        this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
150:        this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
151:        this.ClientSize = new System.Drawing.Size(668, 287);
152:        this.Controls.Add(this.btnCombineITextSharp);
153:        this.Controls.Add(this.pbFiles);
154:        this.Controls.Add(this.lblDetails);
155:        this.Controls.Add(this.lbFiles);
156:        this.Controls.Add(this.btnClearList);
157:        this.Controls.Add(this.btnAddFile);
158:        this.Controls.Add(this.btnAddFolder);
159:        this.Controls.Add(this.btnCombinePdfSharp);
160:        this.Name = "FrmMain";
161:        this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
162:        this.Text = "PDF Combiner";
163:        this.FormClosed += new
System.Windows.Forms.FormClosedEventHandler(FrmMain_FormClosed);
164:        this.Shown += new System.EventHandler(this.FrmMain_Shown);
165:        this.menuStrip.ResumeLayout(false);
```

```
166:        this.ResumeLayout(false);
167:        this.PerformLayout();
168:
169: }
```

1.1.2.3.3.11 FrmMain.InitializeFileDialog Method

This function is used to set options of file dialog like Filter, start path etc.

C#

```
private static void InitializeFileDialog(OpenFileDialog dialogAddFile);
```

Parameters

Parameters	Description
OpenFileDialog dialogAddFile	File Dialog

Body Source

```
1: private static void InitializeFileDialog(OpenFileDialog dialogAddFile)
2: {
3:     dialogAddFile.Multiselect = true;
4:     dialogAddFile.Filter = "pdf files (*.pdf)|*.pdf";
5:     dialogAddFile.InitialDirectory = Application.StartupPath;
6:     dialogAddFile.Title = "Select PDF File/s";
7:     dialogAddFile.DefaultExt = "pdf";
8: }
```

1.1.2.3.3.12 FrmMain.lbFiles_DragDrop Method

This is used to drag and drop an item in listbox Which is used to order books and combine them with that order

C#

```
private void lbFiles_DragDrop(object sender, DragEventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
DragEventArgs e	Event Arguments

Description

m>

Body Source

```
1: private void lbFiles_DragDrop(object sender, DragEventArgs e)
2: {
3:     var point = lbFiles.PointToClient(new Point(e.X, e.Y));
4:     var index = this.lbFiles.IndexFromPoint(point);
5:     if (index < 0) index = this.lbFiles.Items.Count - 1;
6:     // type of string because file names stored in string in listbox
7:     var data = e.Data.GetData(typeof(string));
8:     this.lbFiles.Items.Remove(data);
9:     this.lbFiles.Items.Insert(index, data);
10: }
```

1.1.2.3.3.13 FrmMain.lbFiles_DragOver Method

This is used to give effect while dragging and dropping an item in listbox

C#

```
private static void lbFiles_DragOver(object sender, DragEventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
DragEventArgs e	Event Arguments

Body Source

```
1: private static void lbFiles_DragOver(object sender, DragEventArgs e)
2: {
3:     e.Effect = DragDropEffects.Move;
4: }
```

1.1.2.3.3.14 FrmMain.lbFiles_KeyDown Method

This function is used to delete files in listbox which are chosen If you press the Delete button in your keyboard The selected files will be deleted from the list

C#

```
private void lbFiles_KeyDown(object sender, KeyEventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
KeyEventArgs e	Event Arguments

Body Source

```
1: private void lbFiles_KeyDown(object sender, KeyEventArgs e)
2: {
3:     if (e.KeyValue == Keys.Delete.GetHashCode())
4:         DeleteFilesFromListBox();
5: }
```

1.1.2.3.3.15 FrmMain.lbFiles_MouseDown Method

This is used to control mouse actions and if that is not right click Start to drag and drop item in listbox

C#

```
private void lbFiles_MouseDown(object sender, MouseEventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
MouseEventArgs e	Event Arguments

Body Source

```
1: private void lbFiles_MouseDown(object sender, MouseEventArgs e)
2: {
3:     if (!e.Button.Equals(MouseButtons.Right))
4:     {
5:         if (lbFiles.SelectedItem == null) return;
6:         lbFiles.DoDragDrop(lbFiles.SelectedItem, DragDropEffects.Move);
7:     }
8: }
```

1.1.2.3.3.16 FrmMain.menuItemDelete_Click Method

This is used to delete selected items in listbox When menu item Delete is clicked and opened dialog

C#

```
private void menuItemDelete_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```
 1: private void menuItemDelete_Click(object sender, EventArgs e)
 2: {
 3:     if (lbFiles.Items.Count > 0)
 4:     {
 5:         if (lbFiles.SelectedItems.Count > 0)
 6:         {
 7:             var deleteDialog = new DialogResult();
 8:             deleteDialog = MessageBox.Show("You are about to delete " +
lbFiles.SelectedItems.Count + " files in listbox. Are you sure?", AppTitle,
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
 9:             if (deleteDialog == DialogResult.Yes)
10:                 DeleteFilesFromListBox();
11:         }
12:         else
13:             MessageBox.Show("There is no selected files in list.", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Error);
14:     }
15:     else
16:         MessageBox.Show("There is no files in list.", AppTitle, MessageBoxButtons.OK,
MessageBoxIcon.Error);
17: }
```

1.1.2.3.3.17 FrmMain.menuItemOrderAscending_Click Method

This method is used to order elements in listbox ascending

C#

```
private void menuItemOrderAscending_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)
EventArgs e	Event Arguments

Body Source

```
 1: private void menuItemOrderAscending_Click(object sender, EventArgs e)
 2: {
 3:     if (lbFiles.Items.Count > 0)
 4:         lbFiles = SortItems(lbFiles, true);
 5:     else
 6:         MessageBox.Show("There is no files in list.", AppTitle, MessageBoxButtons.OK,
MessageBoxIcon.Error);
 7: }
```

1.1.2.3.3.18 FrmMain.menuItemOrderDescending_Click Method

This method is used to order elements in listbox descending by name

C#

```
private void menuItemOrderDescending_Click(object sender, EventArgs e);
```

Parameters

Parameters	Description
object sender	The sender info (For example Main Form)

| EventArgs e | Event Arguments |

Body Source

```
1: private void menuItemOrderDescending_Click(object sender, EventArgs e)
2: {
3:     if (lbFiles.Items.Count > 0)
4:         lbFiles = SortItems(lbFiles, false);
5:     else
6:         MessageBox.Show("There is no files in list.", AppTitle, MessageBoxButtons.OK,
MessageBoxIcon.Error);
7: }
```

1.1.2.3.3.19 FrmMain.SortItems Method

This method is used to order items in listbox And return them with the given order as parametre

C#

```
private static ListBox SortItems(ListBox listBox, bool ascending);
```

Parameters

Parameters	Description
ListBox listBox	ListBox Info
bool ascending	Order Type is Ascending Or Not

Body Source

```
 1: private static ListBox SortItems(ListBox listBox, bool ascending)
 2: {
 3:     List<object> items;
 4:     items = listBox.Items.OfType<object>().ToList();
 5:     listBox.Items.Clear();
 6:     if (ascending)
 7:         listBox.Items.AddRange(items.OrderBy(i => i).ToArray());
 8:     else
 9:         listBox.Items.AddRange(items.OrderByDescending(i => i).ToArray());
10:     return listBox;
11: }
```

1.1.2.4 Program Class

Class Hierarchy

PdfCombiner.Program

C#

```
public static class Program;
```

File

Program.cs (see page 37)

Description

This is class PdfCombiner.Program.

Program Methods

	Name	Description
	Main (see page 23)	The main entry point for the application.

1.1.2.4.1 Program Methods

1.1.2.4.1.1 Program.Main Method

The main entry point for the application.

C#

```
[STAThread]
private static void Main();
```

Body Source

```
1: [STAThread]
2: static void Main()
3: {
4:     Application.EnableVisualStyles();
5:     Application.SetCompatibleTextRenderingDefault(false);
6:     Application.Run(new FrmMain());
7: }
```

1.2 Files

The following table lists files in this documentation.

Files

Name	Description
AssemblyInfo.cs (⧉ see page 23)	This is file AssemblyInfo.cs.
BaseException.cs (⧉ see page 24)	This is file BaseException.cs.
ExceptionExtensions.cs (⧉ see page 25)	This is file ExceptionExtensions.cs.
FrmMain.cs (⧉ see page 25)	This is file FrmMain.cs.
FrmMain.Designer.cs (⧉ see page 33)	This is file FrmMain.Designer.cs.
PdfCombiner.csproj (⧉ see page 36)	This is file PdfCombiner.csproj.
PdfCombiner.sln (⧉ see page 36)	This is file PdfCombiner.sln.
Program.cs (⧉ see page 37)	This is file Program.cs.
Resources.Designer.cs (⧉ see page 37)	This code was generated by a tool. Runtime Version:4.0.30319.42000 Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.
Settings.Designer.cs (⧉ see page 38)	This code was generated by a tool. Runtime Version:4.0.30319.42000 Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.

1.2.1 AssemblyInfo.cs

This is file AssemblyInfo.cs.

Body Source

```
1: ?using System.Reflection;
2: using System.Runtime.CompilerServices;
3: using System.Runtime.InteropServices;
4:
5: // General Information about an assembly is controlled through the following
6: // set of attributes. Change these attribute values to modify the information
7: // associated with an assembly.
8: [assembly: AssemblyTitle("PdfCombiner")]
```

```
 9: [assembly: AssemblyDescription("")]
10: [assembly: AssemblyConfiguration("")]
11: [assembly: AssemblyCompany("")]
12: [assembly: AssemblyProduct("PdfCombiner")]
13: [assembly: AssemblyCopyright("Copyright ©  2022")]
14: [assembly: AssemblyTrademark("")]
15: [assembly: AssemblyCulture("")]
16:
17: // Setting ComVisible to false makes the types in this assembly not visible
18: // to COM components.  If you need to access a type in this assembly from
19: // COM, set the ComVisible attribute to true on that type.
20: [assembly: ComVisible(false)]
21:
22: // The following GUID is for the ID of the typelib if this project is exposed to COM
23: [assembly: Guid("48e34ee8-dfaa-48db-acfd-70573cee9c00")]
24:
25: // Version information for an assembly consists of the following four values:
26: //
27: //      Major Version
28: //      Minor Version
29: //      Build Number
30: //      Revision
31: //
32: // You can specify all the values or you can default the Build and Revision Numbers
33: // by using the '*' as shown below:
34: // [assembly: AssemblyVersion("1.0.*")]
35: [assembly: AssemblyVersion("1.0.0.0")]
36: [assembly: AssemblyFileVersion("1.0.0.0")]
```

1.2.2 BaseException.cs

This is file BaseException.cs.

Body Source

```
 1: ?using System;
 2:
 3: namespace PdfCombiner
 4: {
 5:     /// <summary>
 6:     /// This function is used to take exceptions which can be thrown in
 7:     /// PDF Combiner App
 8:     /// </summary>
 9:     [Serializable]
10:     public abstract class BaseException : Exception
11:     {
12:         protected BaseException()
13:         {
14:         }
15:
16:         protected BaseException(string message) : base(message)
17:         {
18:         }
19:
20:         protected BaseException(string message, Exception innerException) :
base(message, innerException)
21:         {
22:         }
23:     }
24: }
```

Namespaces

Name	Description
PdfCombiner (⬚ see page 1)	This is namespace PdfCombiner.

1.2.3 ExceptionExtensions.cs

This is file ExceptionExtensions.cs.

Body Source

```
 1: ?using System;
 2:
 3: namespace PdfCombiner
 4: {
 5:     public static class ExceptionExtensions
 6:     {
 7:         /// <summary>
 8:         /// This method is used to take all messages with stack trace info in exception
 9:         /// </summary>
10:         /// <param name="exception">Exception</param>
11:         /// <returns>Detailed Error Message</returns>
12:         public static string GetAllMessages(this Exception exception)
13:         {
14:             var messages = exception.Message;
15:             if (exception.InnerException != null)
16:                 messages += " " + exception.InnerException.GetAllMessages();
17:             if (!string.IsNullOrEmpty(exception.StackTrace))
18:                 messages += " " + exception.StackTrace;
19:             if (!string.IsNullOrEmpty(exception.Source))
20:                 messages += " " + exception.Source;
21:             return messages;
22:         }
23:     }
24: }
```

Namespaces

Name	Description
PdfCombiner (⊡ see page 1)	This is namespace PdfCombiner.

1.2.4 FrmMain.cs

This is file FrmMain.cs.

Body Source

```
 1: ?using iTextSharp.text;
 2: using PdfSharp.Pdf;
 3: using PdfSharp.Pdf.IO;
 4: using System;
 5: using System.Collections.Generic;
 6: using System.Drawing;
 7: using System.IO;
 8: using System.Linq;
 9: using System.Windows.Forms;
10:
11: namespace PdfCombiner
12: {
13:     public partial class FrmMain : Form
14:     {
15:         public const string AppTitle = "PDF Combiner";
16:
17:         public FrmMain()
18:         {
19:             InitializeComponent();
```

```
20:              }
21:
22:          #region Form Init And Finalize Methods
23:
24:          /// <summary>
25:          /// This function is used to terminate application
26:          /// </summary>
27:          /// <param name="sender">The sender info (For example Main Form)</param>
28:          /// <param name="e">Event Arguments</param>
29:          private static void FrmMain_FormClosed(object sender, FormClosedEventArgs e)
30:          {
31:              MessageBox.Show("Thanks for using this app", AppTitle,
     MessageBoxButtons.OK, MessageBoxIcon.Information);
32:              Application.Exit();
33:          }
34:
35:          /// <summary>
36:          /// This function is used to assign menu to listbox
37:          /// When form is shown
38:          /// </summary>
39:          /// <param name="sender">The sender info (For example Main Form)</param>
40:          /// <param name="e">Event Arguments</param>
41:          private void FrmMain_Shown(object sender, EventArgs e)
42:          {
43:              lbFiles.ContextMenuStrip = menuStrip;
44:          }
45:
46:          #endregion
47:
48:          #region Add Item Methods
49:
50:          /// <summary>
51:          /// It is used to add single or multiple PDF files to combine
52:          /// When you choose file or files
53:          /// The listbox in the form will be filled with the name of the files
54:          /// Which you choose to combine
55:          /// </summary>
56:          /// <param name="sender">The sender info (For example Main Form)</param>
57:          /// <param name="e">Event Arguments</param>
58:          private void btnAddFile_Click(object sender, EventArgs e)
59:          {
60:              using (var dialogAddFile = new OpenFileDialog())
61:              {
62:                  InitializeFileDialog(dialogAddFile);
63:                  var result = dialogAddFile.ShowDialog();
64:                  var addedFileCount = 0;
65:                  if (result == DialogResult.OK && dialogAddFile.FileNames != null)
66:                  {
67:                      foreach (var file in dialogAddFile.FileNames)
68:                      {
69:                          if (!lbFiles.Items.Contains(file))
70:                          {
71:                              lbFiles.Items.Add(file);
72:                              addedFileCount++;
73:                          }
74:                      }
75:                      MessageBox.Show(addedFileCount + " file/s successfully added to
     the list", AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
76:                  }
77:              }
78:          }
79:
80:          /// <summary>
81:          /// This function is used to set options of file dialog like
82:          /// Filter, start path etc.
83:          /// </summary>
84:          /// <param name="dialogAddFile">File Dialog</param>
85:          private static void InitializeFileDialog(OpenFileDialog dialogAddFile)
86:          {
```

```
87:                   dialogAddFile.Multiselect = true;
88:                   dialogAddFile.Filter = "pdf files (*.pdf)|*.pdf";
89:                   dialogAddFile.InitialDirectory = Application.StartupPath;
90:                   dialogAddFile.Title = "Select PDF File/s";
91:                   dialogAddFile.DefaultExt = "pdf";
92:               }
93:
94:           /// <summary>
95:           /// This function is used to add PDF files in a folder which you choose in
96:           /// Folder Dialog recursively
97:           /// When you use it the added files will be seen in listbox
98:           /// </summary>
99:           /// <param name="sender">The sender info (For example Main Form)</param>
100:          /// <param name="e">Event Arguments</param>
101:          private void btnAddFolder_Click(object sender, EventArgs e)
102:          {
103:              using (var dialogAddFolder = new FolderBrowserDialog())
104:              {
105:                  var result = dialogAddFolder.ShowDialog();
106:                  if (result == DialogResult.OK &&
!string.IsNullOrEmpty(dialogAddFolder.SelectedPath))
107:                  {
108:                      var fileNames = Directory.GetFiles(dialogAddFolder.SelectedPath,
"*.pdf", SearchOption.AllDirectories);
109:                      var addedFileCount = 0;
110:                      foreach (var file in fileNames)
111:                      {
112:                          if (!lbFiles.Items.Contains(file))
113:                          {
114:                              lbFiles.Items.Add(file);
115:                              addedFileCount++;
116:                          }
117:                      }
118:                      MessageBox.Show(addedFileCount + " file/s successfully added to
the list", AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
119:                  }
120:              }
121:          }
122:
123:          #endregion
124:
125:          #region Combine PDF Methods
126:
127:          /// <summary>
128:          /// This function is used to take list of PDF files in listbox,
129:          /// Combine them with a single file in location which you choose in folder
dialog
130:          /// And you can see the progress in progress bar
131:          /// It uses PdfSharp Nuget Package
132:          /// </summary>
133:          /// <param name="sender">The sender info (For example Main Form)</param>
134:          /// <param name="e">Event Arguments</param>
135:          private void btnCombinePdfSharp_Click(object sender, EventArgs e)
136:          {
137:              try
138:              {
139:                  if (lbFiles.Items.Count < 2)
140:                      MessageBox.Show("There must be at least 2 PDF files to combine",
AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Error);
141:                  else
142:                  {
143:                      using (var dialogExport = new FolderBrowserDialog())
144:                      {
145:                          var result = dialogExport.ShowDialog();
146:                          if (result == DialogResult.OK &&
!string.IsNullOrEmpty(dialogExport.SelectedPath))
147:                          {
148:                              var outputFileName = dialogExport.SelectedPath + "/" +
Guid.NewGuid() + ".pdf";
```

```
149:                                using (var outputFile = new PdfDocument())
150:                                {
151:                                        outputFile.Options.CompressContentStreams = true;
152:
outputFile.Options.EnableCcittCompressionForBilevelImages = true;
153:                                        outputFile.Options.FlateEncodeMode =
PdfFlateEncodeMode.BestCompression;
154:                                    var fileCount = lbFiles.Items.Count;
155:                                    var combinedFiles = 0;
156:
157:                                    for (int i = 0; i < lbFiles.Items.Count; i++)
158:                                    {
159:                                        try
160:                                        {
161:                                            var inputDocument =
PdfReader.Open(lbFiles.Items[i].ToString(), PdfDocumentOpenMode.Import);
162:                                            var count = inputDocument.PageCount;
163:                                            for (int idx = 0; idx < count; idx++)
164:                                            {
165:                                                var page = inputDocument.Pages[idx];
166:                                                outputFile.AddPage(page);
167:                                            }
168:                                            inputDocument.Close();
169:                                            combinedFiles++;
170:                                        }
171:                                        catch (Exception)
172:                                        {
173:                                            fileCount--;
174:                                        }
175:
176:                                        pbFiles.Value = (combinedFiles * 100 / fileCount);
177:                                        if (pbFiles.Value > pbFiles.Maximum)
178:                                            pbFiles.Value = pbFiles.Maximum;
179:                                        ActiveForm.Text = "%" + pbFiles.Value;
180:                                    }
181:                                    outputFile.Save(outputFileName);
182:                                    outputFile.Close();
183:
184:                                    MessageBox.Show(fileCount + " PDF files successfully
combined in " + outputFileName, AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
185:                                }
186:
187:                                pbFiles.Value = pbFiles.Minimum;
188:                                ActiveForm.Text = AppTitle;
189:                            }
190:                        }
191:                    }
192:                }
193:            catch (Exception ex)
194:            {
195:                MessageBox.Show("There is an error while combining PDF files in list.
Details: " + ex.GetAllMessages(), AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Error);
196:                pbFiles.Value = pbFiles.Minimum;
197:                ActiveForm.Text = AppTitle;
198:            }
199:        }
200:
201:        /// <summary>
202:        /// This function is used to take list of PDF files in listbox,
203:        /// Combine them with a single file in location which you choose in folder
dialog
204:        /// And you can see the progress in progress bar
205:        /// It uses iTextSharp Nuget Package
206:        /// </summary>
207:        /// <param name="sender">The sender info (For example Main Form)</param>
208:        /// <param name="e">Event Arguments</param>
209:        private void btnCombineITextSharp_Click(object sender, EventArgs e)
210:        {
211:            try
```

```
212:                        {
213:                            if (lbFiles.Items.Count < 2)
214:                                MessageBox.Show("There must be at least 2 PDF files to combine",
AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Error);
215:                            else
216:                            {
217:                                using (var dialogExport = new FolderBrowserDialog())
218:                                {
219:                                    var result = dialogExport.ShowDialog();
220:                                    if (result == DialogResult.OK &&
!string.IsNullOrEmpty(dialogExport.SelectedPath))
221:                                    {
222:                                        var outputFileName = dialogExport.SelectedPath + "/" +
Guid.NewGuid() + ".pdf";
223:                                        var fileCount = lbFiles.Items.Count;
224:                                        var combinedFiles = 0;
225:
226:                                        var outputFile = new Document();
227:                                        using (FileStream outputFileStream = new
FileStream(outputFileName, FileMode.Create))
228:                                        {
229:                                            using (var pdfWriter = new
iTextSharp.text.pdf.PdfCopy(outputFile, outputFileStream))
230:                                            {
231:                                                if (pdfWriter == null)
232:                                                {
233:                                                    return;
234:                                                }
235:                                                pdfWriter.SetFullCompression();
236:                                                outputFile.Open();
237:                                                for (int i = 0; i < lbFiles.Items.Count; i++)
238:                                                {
239:                                                    try
240:                                                    {
241:                                                        var pdfReader = new
iTextSharp.text.pdf.PdfReader(lbFiles.Items[i].ToString());
242:                                                        pdfReader.ConsolidateNamedDestinations();
243:                                                        for (int j = 1; j <=
pdfReader.NumberOfPages; j++)
244:                                                        {
245:                                                            var page =
pdfWriter.GetImportedPage(pdfReader, j);
246:                                                            pdfWriter.AddPage(page);
247:                                                        }
248:                                                        pdfReader.Close();
249:                                                        combinedFiles++;
250:                                                    }
251:                                                    catch (Exception)
252:                                                    {
253:                                                        fileCount--;
254:                                                    }
255:
256:                                                    pbFiles.Value = (combinedFiles * 100 /
fileCount);
257:                                                    if (pbFiles.Value > pbFiles.Maximum)
258:                                                        pbFiles.Value = pbFiles.Maximum;
259:                                                    ActiveForm.Text = "%" + pbFiles.Value;
260:                                                }
261:                                                pdfWriter.Close();
262:                                            }
263:                                            outputFile.Close();
264:                                        }
265:
266:                                        MessageBox.Show(fileCount + " PDF files successfully
combined in " + outputFileName, AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
267:
268:                                        pbFiles.Value = pbFiles.Minimum;
269:                                        ActiveForm.Text = AppTitle;
270:                                    }
```

```
271:                              }
272:                          }
273:                      }
274:                  catch (Exception ex)
275:                  {
276:                      MessageBox.Show("There is an error while combining PDF files in list.
Details: " + ex.GetAllMessages(), AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Error);
277:                      pbFiles.Value = pbFiles.Minimum;
278:                      ActiveForm.Text = AppTitle;
279:                  }
280:              }
281:
282:          #endregion
283:
284:          #region Delete Item Methods
285:
286:          /// <summary>
287:          /// This function is used to delete files in listbox which are chosen
288:          /// If you press the Delete button in your keyboard
289:          /// The selected files will be deleted from the list
290:          /// </summary>
291:          /// <param name="sender">The sender info (For example Main Form)</param>
292:          /// <param name="e">Event Arguments</param>
293:          private void lbFiles_KeyDown(object sender, KeyEventArgs e)
294:          {
295:              if (e.KeyValue == Keys.Delete.GetHashCode())
296:                  DeleteFilesFromListBox();
297:          }
298:
299:          /// <summary>
300:          /// This is used to delete selected items in listbox
301:          /// When menu item Delete is clicked and opened dialog
302:          /// </summary>
303:          /// <param name="sender">The sender info (For example Main Form)</param>
304:          /// <param name="e">Event Arguments</param>
305:          private void menuItemDelete_Click(object sender, EventArgs e)
306:          {
307:              if (lbFiles.Items.Count > 0)
308:              {
309:                  if (lbFiles.SelectedItems.Count > 0)
310:                  {
311:                      var deleteDialog = new DialogResult();
312:                      deleteDialog = MessageBox.Show("You are about to delete " +
lbFiles.SelectedItems.Count + " files in listbox. Are you sure?", AppTitle,
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
313:                      if (deleteDialog == DialogResult.Yes)
314:                          DeleteFilesFromListBox();
315:                  }
316:                  else
317:                      MessageBox.Show("There is no selected files in list.", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Error);
318:              }
319:              else
320:                  MessageBox.Show("There is no files in list.", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Error);
321:          }
322:
323:          /// <summary>
324:          /// This method is used to Delete selected files from ListBox
325:          /// This is used different ways in this app
326:          /// So we created this as a method
327:          /// </summary>
328:          private void DeleteFilesFromListBox()
329:          {
330:              var allItems = new List<string>();
331:              var removedItems = new List<string>();
332:              var fileCount = lbFiles.SelectedItems.Count;
333:              for (int i = 0; i < lbFiles.SelectedItems.Count; i++)
334:              {
```

```
335:                         removedItems.Add(lbFiles.SelectedItems[i].ToString());
336:                     }
337:                 foreach (var item in lbFiles.Items)
338:                 {
339:                     if (!removedItems.Any(j => j == item.ToString()))
340:                         allItems.Add(item.ToString());
341:                 }
342:                 lbFiles.Items.Clear();
343:                 foreach (var item in allItems)
344:                 {
345:                     lbFiles.Items.Add(item);
346:                 }
347:                 MessageBox.Show(fileCount + " file/s are deleted from the list.",
AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
348:             }
349:
350:         /// <summary>
351:         /// This function is used to clear the file list in listbox
352:         /// </summary>
353:         /// <param name="sender">The sender info (For example Main Form)</param>
354:         /// <param name="e">Event Arguments</param>
355:         private void btnClearList_Click(object sender, EventArgs e)
356:         {
357:             var fileCount = lbFiles.Items.Count;
358:             lbFiles.Items.Clear();
359:             MessageBox.Show(fileCount + " file/s successfully deleted from the list",
AppTitle, MessageBoxButtons.OK, MessageBoxIcon.Information);
360:         }
361:
362:         #endregion
363:
364:         #region Order Item Methods
365:
366:         /// <summary>
367:         /// This method is used to order elements in listbox ascending
368:         /// </summary>
369:         /// <param name="sender">The sender info (For example Main Form)</param>
370:         /// <param name="e">Event Arguments</param>
371:         private void menuItemOrderAscending_Click(object sender, EventArgs e)
372:         {
373:             if (lbFiles.Items.Count > 0)
374:                 lbFiles = SortItems(lbFiles, true);
375:             else
376:                 MessageBox.Show("There is no files in list.", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Error);
377:         }
378:
379:         /// <summary>
380:         /// This method is used to order elements in listbox descending by name
381:         /// </summary>
382:         /// <param name="sender">The sender info (For example Main Form)</param>
383:         /// <param name="e">Event Arguments</param>
384:         private void menuItemOrderDescending_Click(object sender, EventArgs e)
385:         {
386:             if (lbFiles.Items.Count > 0)
387:                 lbFiles = SortItems(lbFiles, false);
388:             else
389:                 MessageBox.Show("There is no files in list.", AppTitle,
MessageBoxButtons.OK, MessageBoxIcon.Error);
390:         }
391:
392:         /// <summary>
393:         /// This method is used to order items in listbox
394:         /// And return them with the given order as parametre
395:         /// </summary>
396:         /// <param name="listBox">ListBox Info</param>
397:         /// <param name="ascending">Order Type is Ascending Or Not</param>
398:         private static ListBox SortItems(ListBox listBox, bool ascending)
399:         {
```

```
400:                    List<object> items;
401:                    items = listBox.Items.OfType<object>().ToList();
402:                    listBox.Items.Clear();
403:                    if (ascending)
404:                        listBox.Items.AddRange(items.OrderBy(i => i).ToArray());
405:                    else
406:                        listBox.Items.AddRange(items.OrderByDescending(i => i).ToArray());
407:                    return listBox;
408:                }
409:
410:            #endregion
411:
412:            #region Drag Drop Item For Order Methods
413:
414:            /// <summary>
415:            /// This is used to control mouse actions and if that is not right click
416:            /// Start to drag and drop item in listbox
417:            /// </summary>
418:            /// <param name="sender">The sender info (For example Main Form)</param>
419:            /// <param name="e">Event Arguments</param>
420:            private void lbFiles_MouseDown(object sender, MouseEventArgs e)
421:            {
422:                if (!e.Button.Equals(MouseButtons.Right))
423:                {
424:                    if (lbFiles.SelectedItem == null) return;
425:                    lbFiles.DoDragDrop(lbFiles.SelectedItem, DragDropEffects.Move);
426:                }
427:            }
428:
429:            /// <summary>
430:            /// This is used to give effect while dragging and dropping an item in listbox
431:            /// </summary>
432:            /// <param name="sender">The sender info (For example Main Form)</param>
433:            /// <param name="e">Event Arguments</param>
434:            private static void lbFiles_DragOver(object sender, DragEventArgs e)
435:            {
436:                e.Effect = DragDropEffects.Move;
437:            }
438:
439:            /// <summary>
440:            /// This is used to drag and drop an item in listbox
441:            /// Which is used to order books and combine them with that order
442:            /// </summary>
443:            /// <param name="sender">The sender info (For example Main Form)</param>
444:            /// <param name="e">Event Arguments</param>m
445:            private void lbFiles_DragDrop(object sender, DragEventArgs e)
446:            {
447:                var point = lbFiles.PointToClient(new Point(e.X, e.Y));
448:                var index = this.lbFiles.IndexFromPoint(point);
449:                if (index < 0) index = this.lbFiles.Items.Count - 1;
450:                // type of string because file names stored in string in listbox
451:                var data = e.Data.GetData(typeof(string));
452:                this.lbFiles.Items.Remove(data);
453:                this.lbFiles.Items.Insert(index, data);
454:            }
455:            #endregion
456:        }
457: }
```

Namespaces

Name	Description
PdfCombiner (⊡ see page 1)	This is namespace PdfCombiner.

1.2.5 FrmMain.Designer.cs

This is file FrmMain.Designer.cs.

Body Source

```
 1: ?namespace PdfCombiner
 2: {
 3:     partial class FrmMain
 4:     {
 5:         /// <summary>
 6:         /// Required designer variable.
 7:         /// </summary>
 8:         private System.ComponentModel.IContainer components;
 9:
10:         /// <summary>
11:         /// Clean up any resources being used.
12:         /// </summary>
13:         /// <param name="disposing">true if managed resources should be disposed;
otherwise, false.</param>
14:         protected override void Dispose(bool disposing)
15:         {
16:             if (disposing && (components != null))
17:             {
18:                 components.Dispose();
19:             }
20:             base.Dispose(disposing);
21:         }
22:
23:         #region Windows Form Designer generated code
24:
25:         /// <summary>
26:         /// Required method for Designer support - do not modify
27:         /// the contents of this method with the code editor.
28:         /// </summary>
29:         private void InitializeComponent()
30:         {
31:             this.components = new System.ComponentModel.Container();
32:             System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(FrmMain));
33:             this.btnCombinePdfSharp = new System.Windows.Forms.Button();
34:             this.btnAddFolder = new System.Windows.Forms.Button();
35:             this.btnAddFile = new System.Windows.Forms.Button();
36:             this.btnClearList = new System.Windows.Forms.Button();
37:             this.lbFiles = new System.Windows.Forms.ListBox();
38:             this.lblDetails = new System.Windows.Forms.Label();
39:             this.pbFiles = new System.Windows.Forms.ProgressBar();
40:             this.btnCombineITextSharp = new System.Windows.Forms.Button();
41:             this.menuStrip = new
System.Windows.Forms.ContextMenuStrip(this.components);
42:             this.menuItemDelete = new System.Windows.Forms.ToolStripMenuItem();
43:             this.menuItemOrderAscending = new System.Windows.Forms.ToolStripMenuItem();
44:             this.menuItemOrderDescending = new
System.Windows.Forms.ToolStripMenuItem();
45:             this.menuStrip.SuspendLayout();
46:             this.SuspendLayout();
47:             //
48:             // btnCombinePdfSharp
49:             //
50:             this.btnCombinePdfSharp.Image =
((System.Drawing.Image)(resources.GetObject("btnCombinePdfSharp.Image")));
51:             this.btnCombinePdfSharp.ImageAlign =
System.Drawing.ContentAlignment.MiddleLeft;
52:             this.btnCombinePdfSharp.Location = new System.Drawing.Point(442, 5);
53:             this.btnCombinePdfSharp.Name = "btnCombinePdfSharp";
```

```
 54:              this.btnCombinePdfSharp.Size = new System.Drawing.Size(214, 37);
 55:              this.btnCombinePdfSharp.TabIndex = 0;
 56:              this.btnCombinePdfSharp.Text = "Combine Files (PdfSharp)";
 57:              this.btnCombinePdfSharp.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
 58:              this.btnCombinePdfSharp.UseVisualStyleBackColor = true;
 59:              this.btnCombinePdfSharp.Click += new
System.EventHandler(this.btnCombinePdfSharp_Click);
 60:              //
 61:              // btnAddFolder
 62:              //
 63:              this.btnAddFolder.Image =
((System.Drawing.Image)(resources.GetObject("btnAddFolder.Image")));
 64:              this.btnAddFolder.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
 65:              this.btnAddFolder.Location = new System.Drawing.Point(12, 45);
 66:              this.btnAddFolder.Name = "btnAddFolder";
 67:              this.btnAddFolder.Size = new System.Drawing.Size(105, 37);
 68:              this.btnAddFolder.TabIndex = 2;
 69:              this.btnAddFolder.Text = "Add Folder";
 70:              this.btnAddFolder.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
 71:              this.btnAddFolder.UseVisualStyleBackColor = true;
 72:              this.btnAddFolder.Click += new
System.EventHandler(this.btnAddFolder_Click);
 73:              //
 74:              // btnAddFile
 75:              //
 76:              this.btnAddFile.Image =
((System.Drawing.Image)(resources.GetObject("btnAddFile.Image")));
 77:              this.btnAddFile.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
 78:              this.btnAddFile.Location = new System.Drawing.Point(12, 5);
 79:              this.btnAddFile.Name = "btnAddFile";
 80:              this.btnAddFile.Size = new System.Drawing.Size(105, 37);
 81:              this.btnAddFile.TabIndex = 3;
 82:              this.btnAddFile.Text = "Add Files";
 83:              this.btnAddFile.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
 84:              this.btnAddFile.UseVisualStyleBackColor = true;
 85:              this.btnAddFile.Click += new System.EventHandler(this.btnAddFile_Click);
 86:              //
 87:              // btnClearList
 88:              //
 89:              this.btnClearList.Image =
((System.Drawing.Image)(resources.GetObject("btnClearList.Image")));
 90:              this.btnClearList.ImageAlign = System.Drawing.ContentAlignment.MiddleLeft;
 91:              this.btnClearList.Location = new System.Drawing.Point(123, 5);
 92:              this.btnClearList.Name = "btnClearList";
 93:              this.btnClearList.Size = new System.Drawing.Size(105, 37);
 94:              this.btnClearList.TabIndex = 4;
 95:              this.btnClearList.Text = "Clear File List";
 96:              this.btnClearList.TextAlign = System.Drawing.ContentAlignment.MiddleRight;
 97:              this.btnClearList.UseVisualStyleBackColor = true;
 98:              this.btnClearList.Click += new
System.EventHandler(this.btnClearList_Click);
 99:              //
100:              // lbFiles
101:              //
102:              this.lbFiles.AllowDrop = true;
103:              this.lbFiles.FormattingEnabled = true;
104:              this.lbFiles.Location = new System.Drawing.Point(12, 88);
105:              this.lbFiles.Name = "lbFiles";
106:              this.lbFiles.SelectionMode =
System.Windows.Forms.SelectionMode.MultiExtended;
107:              this.lbFiles.Size = new System.Drawing.Size(644, 186);
108:              this.lbFiles.TabIndex = 5;
109:              this.lbFiles.DragDrop += new
System.Windows.Forms.DragEventHandler(this.lbFiles_DragDrop);
110:              this.lbFiles.DragOver += new
System.Windows.Forms.DragEventHandler(lbFiles_DragOver);
111:              this.lbFiles.KeyDown += new
System.Windows.Forms.KeyEventHandler(this.lbFiles_KeyDown);
```

```
112:                 this.lbFiles.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.lbFiles_MouseDown);
113:                 //
114:                 // lblDetails
115:                 //
116:                 this.lblDetails.AutoSize = true;
117:                 this.lblDetails.Location = new System.Drawing.Point(12, 65);
118:                 this.lblDetails.Name = "lblDetails";
119:                 this.lblDetails.Size = new System.Drawing.Size(0, 13);
120:                 this.lblDetails.TabIndex = 6;
121:                 //
122:                 // pbFiles
123:                 //
124:                 this.pbFiles.Location = new System.Drawing.Point(123, 45);
125:                 this.pbFiles.Name = "pbFiles";
126:                 this.pbFiles.Size = new System.Drawing.Size(313, 37);
127:                 this.pbFiles.TabIndex = 7;
128:                 //
129:                 // btnCombineITextSharp
130:                 //
131:                 this.btnCombineITextSharp.Image =
((System.Drawing.Image)(resources.GetObject("btnCombineITextSharp.Image")));
132:                 this.btnCombineITextSharp.ImageAlign =
System.Drawing.ContentAlignment.MiddleLeft;
133:                 this.btnCombineITextSharp.Location = new System.Drawing.Point(442, 45);
134:                 this.btnCombineITextSharp.Name = "btnCombineITextSharp";
135:                 this.btnCombineITextSharp.Size = new System.Drawing.Size(214, 37);
136:                 this.btnCombineITextSharp.TabIndex = 9;
137:                 this.btnCombineITextSharp.Text = "Combine Files (iTextSharp)";
138:                 this.btnCombineITextSharp.TextAlign =
System.Drawing.ContentAlignment.MiddleRight;
139:                 this.btnCombineITextSharp.UseVisualStyleBackColor = true;
140:                 this.btnCombineITextSharp.Click += new
System.EventHandler(this.btnCombineITextSharp_Click);
141:                 //
142:                 // menuStrip
143:                 //
144:                 this.menuStrip.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {
145:                 this.menuItemDelete,
146:                 this.menuItemOrderAscending,
147:                 this.menuItemOrderDescending});
148:                 this.menuStrip.Name = "menuStrip";
149:                 this.menuStrip.Size = new System.Drawing.Size(178, 70);
150:                 //
151:                 // menuItemDelete
152:                 //
153:                 this.menuItemDelete.Image =
((System.Drawing.Image)(resources.GetObject("menuItemDelete.Image")));
154:                 this.menuItemDelete.Name = "menuItemDelete";
155:                 this.menuItemDelete.Size = new System.Drawing.Size(177, 22);
156:                 this.menuItemDelete.Text = "Delete";
157:                 this.menuItemDelete.Click += new
System.EventHandler(this.menuItemDelete_Click);
158:                 //
159:                 // menuItemOrderAscending
160:                 //
161:                 this.menuItemOrderAscending.Image =
((System.Drawing.Image)(resources.GetObject("menuItemOrderAscending.Image")));
162:                 this.menuItemOrderAscending.Name = "menuItemOrderAscending";
163:                 this.menuItemOrderAscending.Size = new System.Drawing.Size(177, 22);
164:                 this.menuItemOrderAscending.Text = "Order (Ascending)";
165:                 this.menuItemOrderAscending.Click += new
System.EventHandler(this.menuItemOrderAscending_Click);
166:                 //
167:                 // menuItemOrderDescending
168:                 //
169:                 this.menuItemOrderDescending.Image =
((System.Drawing.Image)(resources.GetObject("menuItemOrderDescending.Image")));
170:                 this.menuItemOrderDescending.Name = "menuItemOrderDescending";
```

```
171:                this.menuItemOrderDescending.Size = new System.Drawing.Size(177, 22);
172:                this.menuItemOrderDescending.Text = "Order (Descending)";
173:                this.menuItemOrderDescending.Click += new
System.EventHandler(this.menuItemOrderDescending_Click);
174:                //
175:                // FrmMain
176:                //
177:                this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
178:                this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
179:                this.ClientSize = new System.Drawing.Size(668, 287);
180:                this.Controls.Add(this.btnCombineITextSharp);
181:                this.Controls.Add(this.pbFiles);
182:                this.Controls.Add(this.lblDetails);
183:                this.Controls.Add(this.lbFiles);
184:                this.Controls.Add(this.btnClearList);
185:                this.Controls.Add(this.btnAddFile);
186:                this.Controls.Add(this.btnAddFolder);
187:                this.Controls.Add(this.btnCombinePdfSharp);
188:                this.Name = "FrmMain";
189:                this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
190:                this.Text = "PDF Combiner";
191:                this.FormClosed += new
System.Windows.Forms.FormClosedEventHandler(FrmMain_FormClosed);
192:                this.Shown += new System.EventHandler(this.FrmMain_Shown);
193:                this.menuStrip.ResumeLayout(false);
194:                this.ResumeLayout(false);
195:                this.PerformLayout();
196:
197:            }
198:
199:            #endregion
200:
201:            private System.Windows.Forms.Button btnCombinePdfSharp;
202:            private System.Windows.Forms.Button btnAddFolder;
203:            private System.Windows.Forms.Button btnAddFile;
204:            private System.Windows.Forms.Button btnClearList;
205:            private System.Windows.Forms.ListBox lbFiles;
206:            private System.Windows.Forms.Label lblDetails;
207:            private System.Windows.Forms.ProgressBar pbFiles;
208:            private System.Windows.Forms.Button btnCombineITextSharp;
209:            private System.Windows.Forms.ContextMenuStrip menuStrip;
210:            private System.Windows.Forms.ToolStripMenuItem menuItemDelete;
211:            private System.Windows.Forms.ToolStripMenuItem menuItemOrderAscending;
212:            private System.Windows.Forms.ToolStripMenuItem menuItemOrderDescending;
213:        }
214: }
215:
```

Namespaces

Name	Description
PdfCombiner (☑ see page 1)	This is namespace PdfCombiner.

1.2.6 **PdfCombiner.csproj**

This is file PdfCombiner.csproj.

1.2.7 **PdfCombiner.sln**

This is file PdfCombiner.sln.

1.2.8 **Program.cs**

This is file Program.cs.

Body Source

```
 1: ?using System;
 2: using System.Collections.Generic;
 3: using System.Linq;
 4: using System.Windows.Forms;
 5:
 6: namespace PdfCombiner
 7: {
 8:     static class Program
 9:     {
10:         /// <summary>
11:         /// The main entry point for the application.
12:         /// </summary>
13:         [STAThread]
14:         static void Main()
15:         {
16:             Application.EnableVisualStyles();
17:             Application.SetCompatibleTextRenderingDefault(false);
18:             Application.Run(new FrmMain());
19:         }
20:     }
21: }
```

Namespaces

Name	Description
PdfCombiner (⬜ see page 1)	This is namespace PdfCombiner.

1.2.9 **Resources.Designer.cs**

This code was generated by a tool. Runtime Version:4.0.30319.42000

Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.

Body Source

```
 1: ?//------------------------------------------------------------------------------
 2: // <auto-generated>
 3: //     This code was generated by a tool.
 4: //     Runtime Version:4.0.30319.42000
 5: //
 6: //     Changes to this file may cause incorrect behavior and will be lost if
 7: //     the code is regenerated.
 8: // </auto-generated>
 9: //------------------------------------------------------------------------------
10:
11: namespace PdfCombiner.Properties {
12:     using System;
13:
14:
15:     /// <summary>
16:     ///   A strongly-typed resource class, for looking up localized strings, etc.
17:     /// </summary>
18:     // This class was auto-generated by the StronglyTypedResourceBuilder
19:     // class via a tool like ResGen or Visual Studio.
20:     // To add or remove a member, edit your .ResX file then rerun ResGen
```

```
21:      // with the /str option, or rebuild your VS project.
22:
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("System.Resources.Tools.StronglyType
dResourceBuilder",
"15.0.0.0")]
23:      [global::System.Diagnostics.DebuggerNonUserCodeAttribute()]
24:      [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
25:      internal class Resources {
26:
27:          private static global::System.Resources.ResourceManager resourceMan;
28:
29:          private static global::System.Globalization.CultureInfo resourceCulture;
30:
31:
[global::System.Diagnostics.CodeAnalysis.SuppressMessageAttribute("Microsoft.Performance",
"CA1811:AvoidUncalledPrivateCode")]
32:          internal Resources() {
33:          }
34:
35:          /// <summary>
36:          ///   Returns the cached ResourceManager instance used by this class.
37:          /// </summary>
38:
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
39:          internal static global::System.Resources.ResourceManager ResourceManager {
40:              get {
41:                  if (object.ReferenceEquals(resourceMan, null)) {
42:                      global::System.Resources.ResourceManager temp = new
global::System.Resources.ResourceManager("PdfCombiner.Properties.Resources",
typeof(Resources).Assembly);
43:                      resourceMan = temp;
44:                  }
45:                  return resourceMan;
46:              }
47:          }
48:
49:          /// <summary>
50:          ///   Overrides the current thread's CurrentUICulture property for all
51:          ///   resource lookups using this strongly typed resource class.
52:          /// </summary>
53:
[global::System.ComponentModel.EditorBrowsableAttribute(global::System.ComponentModel.Editor
BrowsableState.Advanced)]
54:          internal static global::System.Globalization.CultureInfo Culture {
55:              get {
56:                  return resourceCulture;
57:              }
58:              set {
59:                  resourceCulture = value;
60:              }
61:          }
62:      }
63: }
```

Namespaces

Name	Description
PdfCombiner (⊠ see page 1)	This is namespace PdfCombiner.

1.2.10 **Settings.Designer.cs**

This code was generated by a tool. Runtime Version:4.0.30319.42000

Changes to this file may cause incorrect behavior and will be lost if the code is regenerated.

Body Source

```
 1: ?//------------------------------------------------------------------------------
 2: // <auto-generated>
 3: //     This code was generated by a tool.
 4: //     Runtime Version:4.0.30319.42000
 5: //
 6: //     Changes to this file may cause incorrect behavior and will be lost if
 7: //     the code is regenerated.
 8: // </auto-generated>
 9: //------------------------------------------------------------------------------
10:
11: namespace PdfCombiner.Properties
12: {
13:
14:     [global::System.Runtime.CompilerServices.CompilerGeneratedAttribute()]
15:
[global::System.CodeDom.Compiler.GeneratedCodeAttribute("Microsoft.VisualStudio.Editors.Sett
ingsDesigner.SettingsSingleFileGenerator",
"11.0.0.0")]
16:     internal sealed partial class Settings :
global::System.Configuration.ApplicationSettingsBase
17:     {
18:
19:         private static Settings defaultInstance =
((Settings)(global::System.Configuration.ApplicationSettingsBase.Synchronized(new
Settings()))));
20:     }
21: }
```

Namespaces

Name	Description
PdfCombiner (⬚ see page 1)	This is namespace PdfCombiner.

Index