RasterEdge.XDoc.PDF for .NET SDK

Developer's Guide

Getting Started

System Requirements for .NET

Supported Operating System

The following Microsoft Windows operating systems are supported:
- Microsoft Windows XP Home Edition
- Microsoft Windows XP Professional Edition
- Microsoft Windows XP Professional x64 Edition
- Microsoft Windows 2003 Server
- Microsoft Windows 2008 Server R2
- Microsoft Windows Vista
- Microsoft Windows Vista x64 Edition
- Microsoft Windows 7
- Microsoft Windows 7 Enterprise x64 Edition
- Microsoft Windows 7 Professional x64 Edition
- Microsoft Windows 2012 Server x64 Edition
- Microsoft Windows 8
- Microsoft Windows 10

Development Environments

You can use RasterEdge.XDoc.PDF for .NET to develop applications in any development environment that targets the .NET platform, but the following environments are explicitly supported:
- Microsoft Visual Studio 2005
- Microsoft Visual Studio 2008
- Microsoft Visual Studio 2010
- Microsoft Visual Studio 2011
- Microsoft Visual Studio 2012
- Microsoft Visual Studio 2013
- Microsoft Visual Studio 2015

.NET Framework versions supported

The following .NET Framework versions are supported:
- .NET Framework 2.0
- .NET Framework 3.0
- .NET Framework 3.5
- .NET Framework 4.0
- .NET Framework 4.5
- .NET Framework 4.5.1
- .NET Framework 4.5.2

- .NET Framework 4.6

Reference RasterEdge.XDoc.PDF in .NET project

Necessary Libraries

To use RasterEdge.XDoc.PDF library successfully, the following libraries are necessary:
- RasterEdge.Imaging.Basic.dll
- RasterEdge.Imaging.Basic.Codec.dll
- RasterEdge.Imaging.Font.dll
- RasterEdge.Imaging.Drawing.dll
- RasterEdge.Imaging.Processing.dll
- RasterEdge.XImage.Raster.Core.dll
- RasterEdge.XImage.Raster.dll
- RasterEdge.XDoc.PDF.dll

Add References

The following steps will show you how to use in Visual Studio .NET:
1. In The Solution Explorer, expand the project node you want to add a reference to.
2. Right-click the project's **References** node and select **Add Reference**.
3. In the Add Reference dialog box, Click **Browse** and Navigate to the specified folder.
4. Select the dlls as listed in the following screenshot, Click **OK**.
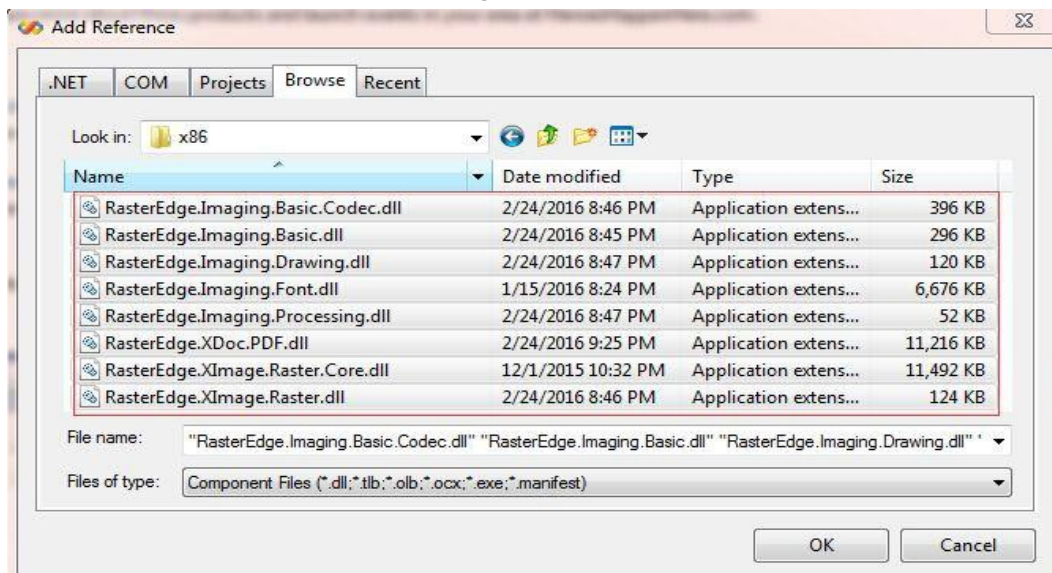
5. The RasterEdge.XDoc.Pdf for .NET reference appears under the project's **References** node.

If you want to know how to select dlls according to your specific development environment, please refer to the **Readme.txt** file in the **/Bin** directory.

FAQ

Errors On Visual Studio

If you get the error as follows:

"Could not load file or assembly 'RasterEdge.XDoc.PDF 'or one of its dependencies. A n attempt was made to load a program with an incorrect format."

Please check your project configures as following ways:

1. If you are using the .NET Framework 4.0 dlls, please confirm that:
    Right-click the project -> Properties ->
    a. Application -> Target framework: .NET Framework 4 or higher
    b. Build -> Platform target: x86 if using x86 dlls, x64 if using x64.
2. If you are using the .NET Framework 2.0 dlls, please confirm that:
    Right-click the project -> Properties ->
    c. Application -> Target framework: .NET Framework 3.0 or 3.5
    d. Build -> Platform target: x86 if using x86 dlls, x64 if using x64.

Errors On IIS

If you configure IIS to run and 500.19 error occurs, then it may be caused by:

1. Not registered the .net framework to the iis. (One of reasons: install a .net framework before the installation of iis.)
2. The site configured in IIS has no sufficient authority to operate. (Modify permission)

There are some solutions:

1. cd to C:\Windows\Microsoft.NET\Framework64\v2.0.50727,Command to re-register net framework to the iis:aspnet_regiis-i.
2. Right-click the correspond site-> Edit Permissions -> Security -> Group or user names-> Edit -> Add -> Add Everyone usersgiven Full Control permissions.

If you get the error as follows:

"Could not load file or assembly "RasterEdge.Imaging.Basic" or any other one assembly or one of its dependencies. An attempt was made to load a program with an incorrect format."

Please check your IIS configure as following ways:

a. If you are using the .NET framework 4.0 or higher dlls, confirm that Web.config is using the content in **Web(for .net4.0 or higher).Config file**.
b. After checking first step, if you are still facing the issue, confirm that:
    If you are using x64 dlls, "Application Pools" -> "Set Application Pool Defaults… "->" Enable 32-Bit Applications" should be false.

If you are using x86 dlls, "Application Pools" -> "Set Application Pool Defaults…" -> "Enable 32-Bit Applications" should be true.

Supported PDF Versions

RasterEdge.XDoc.PDF for .NET supports PDF version 1.2, 1.3, 1.4, 1.5, 1.6 and 1.7

Feature List

Fonts

- 14 core fonts
- Type 1 fonts
- TrueType fonts
- Type 3 fonts
- CJK fonts
- Unicode support

Text

- Add char into PDF page(s)
- Delete char from PDF page(s)
- Add text into PDF page(s)
- Delete text from PDF page specified position
- Extract text from page(s)
- Search text from page(s)
- Replace text on specified page(s)
- Highlight specified text in PDF file

Image

- Add image into PDF file
- Select image(s)
- Extract image(s)
- Delete image(s)
- Replace image(s)

Conversion

- PDF to JPEG (particular page or all pages)
- PDF to JPEG2000 (particular page or all pages)
- PDF to PNG (particular page or all pages)
- PDF to BMP (particular page or all pages)
- PDF to TIFF (particular page or all pages)
- PDF to Gif (particular page or all pages)
- PDF to XPS (particular page or all pages)
- PDF to SVG (particular page or all pages)
- PDF to HTML (particular page or all pages)
- PDF to Word (.docx)
- PDF to Text

PDF Generator

- Extract PDF document from PDF
- Create empty file
- Create PDF from office
- Create PDF from open office
- Create PDF from Tiff
- Create PDF from CSV
- Create PDF from RTF
- Create PDF from Text

Page

- Get page properties
- Convert to Image
- Convert to Vector Image

Header/Footer

- Add Header/Footer
- Remove/Delete Header/Footer

Watermark

- Add Watermark
- Remove/Delete Watermark

Document

- Get page count
- Get document properties
- Insert empty page(s) into a PDF file
- Insert PDF Page(s)
- Delete PDF Page(s)
- Get a particular page
- Replace PDF page(s)
- Swap two pages
- Sort/Reorder PDF pages
- Extract PDF page(s)
- Duplicate page(s)
- Rotate page(s)

Split PDF File

- Split PDF file by page number
- Split PDF file by file size
- Split PDF file by bookmark
- Split PDF file by page index

Combine PDF Files

- Combine PDF objects
- Combine PDF files
- Append PDF file

Annotations

- Annotation Type
- Add annotations
- Get annotations
- Delete annotations
- Modify annotations

Forms

- Add/Insert fields
- Extract field values
- Delete fields
- Modify/Update fields
- Fill field values

Security and Signatures

- Set file permissions
- Modify file password
- Encrypt PDF
- Decrypt PDF
- Edit digital signatures
- Redact text content
- Redact images
- Redact pages

Stamp Annotation

- Create a Stamp Template
- Add Stamp Annotation

Bookmarks

- Extract/Get bookmarks
- Modify bookmarks
- Add/Insert bookmarks
- Delete bookmarks

Printing

- Print PDF to default printer
- Print PDF to a specified printer
- Print PDF to physical or virtual printer
- Print PDF to XPS printer
- Hide print dialog while printing PDF

Compression or Optimizing

- Reduce image size
- Remove unused fonts
- Delete unused fields

Programmer Guide

Working with Text

By using RasterEdge.XDoc.PDF, developer can add/delete a char value, extract, search, replace, delete and highlight specified text into/from PDF file without installing any PDF application freely.

Add char into PDF page(s)

To add a char to an existing PDF file, you may need use as follows:

1. Open the input PDF using the PDFDocument object.
2. Get the PDFTextMgr object of the input PDF object through PDFTextHandler.
3. Add the char value by using method AddChar with specified font, specified page and position.
4. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

The following code will show you how to add a char in an existing PDF file.

C#

```csharp
// Open a document.
String inputFilePath = "C:\\input.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  get a text manager from the document object
PDFTextMgr textMgr = PDFTextHandler.ExportPDFTextManager(doc);
// Set char value.
char aChar = 'A';
// Set text font.
Font font = new Font("Arial", 36F, FontStyle.Regular);
// Get the first page from the document.
int pageIndex = 0;
// Move cursor to (400F, 100F).
PointF cursor = new PointF(400F, 100F);
// Add a character to the page.
textMgr.AddChar(aChar, font, pageIndex, cursor);
// Output the new document.
String outputFilePath = "C:\\output.pdf";
doc.Save(outputFilePath);
```

```vbnet
VB.NET

'Open a document.
Dim inputFilePath As String = "C:\\input.pdf"
Dim doc As PDFDocument = new PDFDocument(inputFilePath)
'get a text manager from the document object
Dim textMgr As PDFTextMgr = PDFTextHandler.ExportPDFTextManager(doc)
'Set char value.
Dim aChar As Char = "A"
'Set text font.
Dim font As Font = new Font("Arial", 36F, FontStyle.Regular)
'Get the first page from the document.
Dim pageIndex As Integer = 0
'Move cursor to (400F, 100F).
Dim cursor As PointF = new PointF(400F, 100F)
'Add a character to the page.
textMgr.AddChar(aChar, font, pageIndex, cursor)
'Output the new document.
Dim outputFilePath As String = "C:\\output.pdf"
doc.Save(outputFilePath)
```

If you need any further information or demo code, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-text-edit-insert/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-text-edit-insert/

Related APIs(**PDFTextMgr.cs**):

```csharp
public int AddChar(char value, Font font, int pageIndex, PointF position)
```

Description:

Add a char value at specified page position with specified font.

Parameters:

Name	Description	Valid Value
value	The char will be added into PDF page	0 ~ 65535
font	The font will be used by char.	The font supported by system.
pageIndex	The page index of the PDF page that will be added the char.	0 ~ page number – 1, which is starting at 0
position	The position will be added a char.	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72.

Return:

Error code, return 0 if success.

```
public int AddChar(char value, Font font, PointF position, int pageIndex,
Color fontColor)
```

Description:

    Add a char value at specified page position with specified font and color.

Parameters:

Name	Description	Valid Value
value	The char will be added into PDF page	0 ~ 65535
font	The font will be used by char.	The font supported by system.
pageIndex	The page index of the PDF page that will be added the char.	0 ~ page number - 1
position	The position will be added a char.	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72.
fontColor	The color that the added char will use.	The color supported by system

Return:

    Error code, return 0 if success.

```
public int AddChar(char value, PSType1Font fontType, float fontSize, PointF
position, Color color, int pageIndex)
```

Description:

    Add a char value at specified page position with specified font and color.

Parameters:

Name	Description	Valid Value
value	The char will be added into PDF page	0 ~ 65535
fontType	The font will be used by char.	The num values in the class PSType1Font
fontSize	The font size of the added char	The value is a positive float
position	The position will be added a char.	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72.
color	The color that the added char will use.	The color supported by system
pageIndex	The page index of the PDF page that will be added the char.	0 ~ page number - 1

Return:

    Error code, return 0 if success.

Delete char from PDF page by Position

To delete a char from an existing PDF file page, you may need use as follows:

1. Open the input PDF using the PDFDocument object.
2. Get the PDFTextMgr object of the input PDF object through PDFTextHandler.
3. Set a point and select the char at the point.
4. Call the PDFTextMgr object's DeleteChar method to delete the selected char.
5. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

The following demo code will show how to delete a char from PDF Page by position.

```csharp
C#

// Open a document.
String inputFilePath = "C:\\1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get a text manager from the document object.
PDFTextMgr textMgr = PDFTextHandler.ExportPDFTextManager(doc);
// Get the first page from the document.
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);
// Select char at position (127F, 187F).
PointF cursor = new PointF(127F, 187F);
PDFTextCharacter aChar = textMgr.SelectChar(page, cursor);
// Delete a selected character.
textMgr.DeleteChar(aChar);
// Output the new document.
String outputFilePath = "C:\\output.pdf";
doc.Save(outputFilePath);
```

```vbnet
VB.NET

'Open a document
Dim inputFilePath As String = "C:\\input.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Get a text manager from the document object
Dim textMgr As PDFTextMgr = PDFTextHandler.ExportPDFTextManager(doc)
'Get the first page from the document
Dim pageIndex As Integer = 0
Dim page As PDFPage = doc.GetPage(pageIndex)
'Select char at position (127F, 187F)
Dim cursor As PointF = New PointF(127.0F, 187.0F)
Dim aChar As PDFTextCharacter = textMgr.SelectChar(page, cursor)
'Delete a selected character
textMgr.DeleteChar(aChar)
'Output the new document
Dim outputFilePath As String = "C:\\output.pdf"
doc.Save(outputFilePath)
```

Also, you can refer to our online demo code sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-text-edit-delete/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-text-edit-delete/

Related APIs(**PDFTextMgr.cs**):

```
public int DeleteChar(PDFTextCharacter value)
```

Description:

Delete a char from PDF page at specified position.

Parameters:

Name	Description	Valid Value
value	The char will be deleted from PDF page	The char must exist in the PDF

Return:

Error code, return 0 if success.

Add text into PDF page(s)

To add text string to an existing PDF file, you may need use as follows:

1. Open the input PDF using the PDFDocument object.
2. Get the PDFTextMgr object of the input PDF object through PDFTextHandler.
3. Add the char value by using method AddString with specified font, specified page and position. You can select different AddString method according to your needs.
4. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

The following code will show you how to add a string in an existing PDF file.

```csharp
C#

// Open a document.
String inputFilePath = "C:\\input.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get a text manager from the document object.
PDFTextMgr textMgr = PDFTextHandler.ExportPDFTextManager(doc);
// Set string value.
String msg = "Hello World";
// Set text font.
Font font = new Font("Arial", 36F, FontStyle.Italic);
// Get the first page from the document.
int pageIndex = 0;
// Move cursor to (400F, 100F).
PointF cursor = new PointF(400F, 100F);
// Set font color: red.
Color fontColor = Color.Red;
// Add a string to the page by page index.
textMgr.AddString(msg, font, pageIndex, cursor, fontColor);
// Output the new document.
String outputFilePath = "C:\\output.pdf";
doc.Save(outputFilePath);
```

```vbnet
VB.NET
'Open a document.
Dim inputFilePath As String = "C:\\1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Get a text manager from the document object.
Dim textMgr As PDFTextMgr = PDFTextHandler.ExportPDFTextManager(doc)
'Set string value.
Dim msg As String = "Hello World"
'Set text font.
Dim font As Font = New Font("Arial", 36.0F, FontStyle.Italic)
'Get the first page from the document.
Dim pageIndex As Integer = 0
'Move cursor to (400F, 100F).
Dim cursor As PointF = New PointF(400.0F, 100.0F)
'Set font color: red.
Dim fontColor As Color = Color.Red
'Add a string to the page.
textMgr.AddString(msg, font, pageIndex, cursor, fontColor)
'Output the new document.
Dim outputFilePath As String = "C:\\output.pdf"
doc.Save(outputFilePath)
```

If you want to get more demo codes, please refer to our online demo sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-text-edit-insert/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-text-edit-insert/

Related APIs(**PDFTextMgr.cs**):

```
public int AddString(String value, Font font, int pageIndex, PointF position,
Color fontColor)
```

Description:

Add a char value at specified page position with specified font and color.

Parameters:

Name	Description	Valid Value
value	The string will be added into PDF page	
font	The font will be used by string text.	The font supported by system.
pageIndex	The page index of the PDF page that will be added the char.	0 ~ page number - 1
position	The position will be added a string text.	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72.
fontColor	The color that the added string text will use.	The color supported by system

Error code, return 0 if success.

```
public int AddString(String value, PDFPage page, PSType1Font fontType, float
fontSize, PointF position, Color fontColor)
```

Description:

Add a string at specified page position with specified font and color.

Parameters:

Name	Description	Valid Value
value	The string will be added into PDF page	
fontType	The font will be used by string text.	The num values in the class PSType1Font
page	The PDF page that will be added string text	The page must exist in the PDF
fontSize	The font size of the added string text	The value is a positive float
position	The position will be added a string text.	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72.
color	The color that the added string text will use.	The color supported by system

Return:

Error code, return 0 if success.

Delete text from PDF page(s)

To delete specified text string from an existing PDF file, you may need use as follows:

1. Open the input PDF using the PDFDocument object.
2. Set text search option to specify regular expression usage .
3. Call the PDFDocument object's SearchTextAndDelete method to delete the searched text.
4. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

Now, you can implement the task to delete text from PDF page as follows:

Please refer to the online sites to get more demo codes:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-text-edit-delete/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-text-edit-delete/

Related APIs(**PDFDocument.cs**)

public void SearchTextAndDelete(String matchString, RESearchOption option)

Description:

Delete specified string from all the PDF pages.

Parameters:

Name	Description	Valid Value
matchString	The string will be deleted.	-
option	The search and delete match rules.	-

public void SearchTextAndDelete(String matchString, RESearchOption option, int pageIndex)

Description:

Delete specified string from the specified page by page index.

Parameters:

Name	Description	Valid Value
matchString	The string will be deleted.	-
option	The search and delete match rules.	-
pageIndex	The page index of the specified page will be deleted a string. e.g: The first page is 0.	0 ~ page number – 1, which is starting at 0

```
public void SearchTextAndDelete(String matchString, RESearchOption option,
int pageOffset, int pageCount)
```

Description:

Delete specified string from the specified pages from pageOffse to pageOffset + pageCount - 1.

Parameters:

Name	Description	Valid Value
matchString	The string will be deleted.	-
option	The search and delete match rules.	-
pageOffset	The page index of the first page that will be deleted a string.	0 ~ page number – 1, which is starting at 0
pageCount	The count of pages will be deleted a string.	The value is 1 ~ page number – pageOffset

Extract text from page(s)

To extract text from PDF file page, the following steps will work:

1. Open the input PDF file through PDFDocument object.
2. Get the PDFTextMgr object of the input PDF object through PDFTextHandler.
3. Extract all the lines by calling the PDFTextMgr object's ExtractTextLine method, extract all the words through the PDFTextMgr object's ExtractTextWord method, extract all the chars with the method ExtractTextCharacter.
4. After get the lines/words/characters, you can use them in your own ways.

The following code will show how to extract text from one PDF page.

C#

```csharp
// Open a document.
PDFDocument doc = new PDFDocument(inputFilePath);
PDFTextMgr textMgr = PDFTextHandler.ExportPDFTextManager(doc);
// Extract text content from first page.
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);
// Get all lines in the page.
List<PDFTextLine> lines = textMgr.ExtractTextLine(page);
// Get all words in the page.
List<PDFTextWord> words = textMgr.ExtractTextWord(page);
// Get all characters in the page.
List<PDFTextCharacter> allChar = textMgr.ExtractTextCharacter(page);
```

VB.NET

```vbnet
'Open a document.
Dim doc As PDFDocument = new PDFDocument(inputFilePath)
Dim textMgr As PDFTextMgr = PDFTextHandler.ExportPDFTextManager(doc)
'Extract text content from first page.
Dim pageIndex As Integer = 0
Dim page As PDFPage = (PDFPage)doc.GetPage(pageIndex)
'Get all lines in the page.
Dim lines = textMgr.ExtractTextLine(page)
'Get all words in the page.
Dim words = textMgr.ExtractTextWord(page)
'Get all characters in the page.
Dim allChar = textMgr.ExtractTextCharacter(page)
```

The following code will be how to extract text from all the PDF pages.

```
C#

// Open a document.
PDFDocument doc = new PDFDocument(inputFilePath);
PDFTextMgr textMgr = PDFTextHandler.ExportPDFTextManager(doc);
// Get all lines in the page.
List<PDFTextLine> lines = textMgr.ExtractTextLine();
// Get all words in the page.
List<PDFTextWord> words = textMgr.ExtractTextWord();
// Get all characters in the page.
List<PDFTextCharacter> allChar = textMgr.ExtractTextCharacter();
```

```
VB.NET

'Open a document.
Dim doc As PDFDocument = new PDFDocument(inputFilePath)
Dim textMgr As PDFTextMgr = PDFTextHandler.ExportPDFTextManager(doc)
'Get all lines in the pdf file.
Dim lines = textMgr.ExtractTextLine()
'Get all words in the pdf file.
Dim words = textMgr.ExtractTextWord()
'Get all characters in the pdf file.
Dim allChar = textMgr.ExtractTextCharacter()
```

Also, you can refer to the following sites to have a try:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-text-extract/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-text-extract/

Related API(s)(**PDFTextMgr.cs**):

```
public List<PDFTextLine> ExtractTextLine()
```
Description:
Extract all lines in the PDF file.
Returns:
A list of line objects.

```
public List<PDFTextLine> ExtractTextLine(PDFPage page)
```
Description:
Extract all lines from the PDF page.
Returns:
A list of line objects.

```
public List<PDFTextWord> ExtractTextWord()
```

Description:

Extract all words in the PDF file.

Returns:

A list of word objects.

```
public List<PDFTextWord> ExtractTextWord(PDFPage page)
```

Description:

Extract all words from the PDF page.

Returns:

A list of word objects.

```
public List<PDFTextCharacter> ExtractTextCharacter()
```

Description:

Extract all characters in the PDF file.

Returns:

A list of character objects.

```
public List<PDFTextCharacter> ExtractTextCharacter(PDFPage page)
```

Description:

Extract all characters from the PDF page.

Returns:

A list of character objects.

Search text from page(s)

To search specified text string from an existing PDF file, you may need use as follows:

1. Open the input PDF using the PDFDocument object.
2. Set text search option to specify regular expression usage .
3. Call the PDFDocument object's Search method to search the text.
4. The returned result is a SearchResult object including the text content, position and so on. You can use the information according to your own way.

The following codes will show how to search a text string using our XDoc.PDF product:

Please refer to the following site to get more demo codes:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-text-search/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-text-search/

Related API(s)(**PDFDocument.cs**):

```csharp
public SearchResult Search(String matchString, RESearchOption option)
```

Description:

Search specified string from all the PDF pages.

Parameters:

Name	Description	Valid Value
matchString	The string will be searched.	-
option	The search rules.	-

Returns:

The results of search, it will include a list of SearchResultItem.

```csharp
public SearchResult Search(String matchString, RESearchOption option, int pageIndex)
```

Description:

Search specified string from the specified page by page index.

Parameters:

Name	Description	Valid Value
matchString	The string will be searched.	-
option	The search rules.	-
pageIndex	The page index of the specified page will be searched a string. e.g: The first page is 0.	0 ~ page number – 1, which is starting at 0.

Returns:

The results of search, it will include a list of SearchResultItem.

```
public SearchResult Search(String matchString, RESearchOption option, int
pageOffset, int pageCount)
```

Description:

Search specified string from the specified pages from pageOffse to pageOffset + pageCount - 1.

Parameters:

Name	Description	Valid Value
matchString	The string will be searched.	-
option	The search rules.	-
pageOffset	The page index of the first page that will be deleted a string.	0 ~ page number – 1, which is starting at 0.
pageCount	The count of pages will be searched a string.	The value is 0 ~ page number – pageOffset

Returns:

The results of search, it will include a list of SearchResultItem.

Replace text on specified page(s)

To replace specified text string in an existing PDF file, the following steps is helpful:

1. Open the input PDF file with PDFDocument object.
2. Create a RESearchOption object and set text search option value to specify regular expression usage.
3. Call the PDFDocument object's Replace method to replace the old string with new text.
4. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

The following demo code will illustrate how to replace text:

C#

```csharp
// Open a document.
PDFDocument doc = new PDFDocument(@"C:\input.pdf");
// Set the search options.
RESearchOption option = new RESearchOption();
option.IgnoreCase = true;
option.WholeWord = true;
option.ContextExpansion = 10;
// Replace "RasterEdge" with "Image".
doc.Replace("RasterEdge", "Image", option);
doc.Save(@"C:\output.pdf");
```

VB.NET

```vbnet
'Open a document
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Set the search options
Dim options As RESearchOption = New RESearchOption()
options.IgnoreCase = False
options.WholeWord = True
'Replace "RasterEdge" with "Image"
doc.Replace("RasterEdge", "Image", options)
doc.Save("C:\\output.pdf")
```

The following sites will show you more demo codes to implement the task replacing text:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-text-edit-replace/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-text-edit-replace/

Related API(s)(**PDFDocument.cs**):

```csharp
public void Replace(String oldString, String newString, RESearchOption
option)
```

Description:

Replace old string by new string in the PDF file.

Parameters:

Name	Description	Valid Value
oldString	The old string will be replaced	-
newString	The new string will be used to replace old string	-
option	The replaced string match rules.	-

```csharp
public void Replace(String oldString, String newString, RESearchOption
option, int pageIndex)
```

Description:

Replace old string by new string in the specified PDF page.

Parameters:

Name	Description	Valid Value
oldString	The old string will be replaced	-
newString	The new string will be used to replace old string	-
option	The replaced string match rules.	-
pageIndex	The page index of the page including old string	0 ~ page number – 1, which is starting at 0

```
public void Replace(String oldString, String newString, RESearchOption
option, int pageOffset, int pageCount)
```

Description:

Replace old string by new string in consecutive PDF pages.

Parameters:

Name	Description	Valid Value
oldString	The old string will be replaced	-
newString	The new string will be used to replace old string	-
option	The replaced string match rules.	-
pageOffset	The page index of the page including old string	0 ~ page number – 1, which is starting at 0
pageCount	The count of pages will be replaced a string.	The value is 0 ~ page number – pageOffset

Search and Highlight Text in PDF file

To highlight specified text string from an existing PDF file, you may need use as follows:
1. Open the input PDF using the PDFDocument object.
2. Set text search option to specify regular expression usage .
3. Call the PDFDocument object's SearchTextAndHighlight method to highlight the searched text.
4. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

Now, you can implement the task to highlight text from PDF page as follows:

Please refer to the online sites to get more demo codes:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-annotate-text-highlight/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-annotate-text-highlight/

Related API(s)(**PDFDocument.cs**)

```
public void SearchTextAndHighlight(String matchString, RESearchOption option)
```

Description:

Highlight specified string from all the PDF pages.

Parameters:

Name	Description	Valid Value
matchString	The string will be deleted.	-
option	The search and delete match rules.	-

```
public void SearchTextAndHighlight(String matchString, RESearchOption option, int pageIndex)
```

Description:

Highlight specified string from the specified page by page index.

Parameters:

Name	Description	Valid Value
matchString	The string will be deleted.	-
option	The search and delete match rules.	-
pageIndex	The page index of the specified page will be deleted a string. e.g: The first page is 0.	0 ~ page number - 1

```
public void SearchTextAndHighlight(String matchString, RESearchOption
option, int pageOffset, int pageCount)
```

Description:

Highlight specified string from the specified pages from pageOffset to pageOffset + pageCount - 1.

Parameters:

Name	Description	Valid Value
matchString	The string will be deleted.	-
option	The search and delete match rules.	-
pageOffset	The page index of the first page that will be deleted a string.	0 ~ page number - 1
pageCount	The count of pages will be deleted a string.	The value is 0 ~ page number – pageOffset

Working with Images

Add/Insert image to PDF file

To add an image to an existing PDF file, the following steps will be enough:

1. Open an input PDF file by using PDFDocument object.
2. Load the image that will be added into the PDF file with Bitmap object.
3. Get the PDF page that will be added an image.
4. Call the method AddImage of PDFImageHandler to add an image to the specified page.
5. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

The following code will show how to add image to PDF page by specified position:

```
C#

String inputFilePath = "C:\\input.pdf";
String outputFilePath = "C:\\output.pdf";
// Load a sample image.
Bitmap anImage = new Bitmap("C:\\addImage.png");
// Open a PDF document.
PDFDocument doc = new PDFDocument(inputFilePath);
// Get the first page of PDF document.
PDFPage page = (PDFPage)doc.GetPage(0);
// Set image position in the page: X = 100F, Y = 400F.
PointF position = new PointF(100F, 400F);
// Add image to the page.
PDFImageHandler.AddImage(page, anImage, position);
// Output the new PDF document.
doc.Save(outputFilePath);
```

```
VB.NET

Dim inputFilePath As String = "C:\\input.pdf"
Dim outputFilePath As String = "C:\\output.pdf"
' Load a sample image.
Dim anImage As Bitmap = new Bitmap("C:\\addImage.png")
' Open a PDF document.
Dim doc As PDFDocument = new PDFDocument(inputFilePath)
' Get the first page of PDF document.
Dim page As PDFPage = (PDFPage)doc.GetPage(0)
' Set image position in the page: X = 100F, Y = 400F.
Dim position As PointF = new PointF(100F, 400F)
' Add image to the page.
PDFImageHandler.AddImage(page, anImage, position)
' Output the new PDF document.
doc.Save(outputFilePath)
```

Also, you can refer to the following sites to test the function adding image to PDF file:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-add-image/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-add-image/

Related API(s)(**PDFImageHandler.cs**):

public static int AddImage(PDFDocument doc, int pageIndex, Bitmap bitmap, PDFItemOptions itemOptions)

Description:

Add an image to a PDF page with the specified options.

Parameters:

Name	Description	Valid Value
doc	The PDF file object	Can't be null
pageIndex	The page index of the page that will be added an image	0 ~ page number – 1, which is starting at 0
bitmap	The image data that will be added to the specified page	Can't be null
itemOptions	The options that will be used to add an image	-

Returns:

Error code, return 0 if success.

public static int AddImage(PDFPage page, Bitmap bitmap, PointF position)

Description:

Add an image to a PDF page at the specified position.

Parameters:

Name	Description	Valid Value
page	The PDF page that will be added an image	Can't be null
bitmap	The image data that will be added to the specified page	Can't be null
position	The position of the added image	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidtg() * 72, and the value of y should be 0 to GetHeight() * 72.

public static int AddImage(PDFPage page, Bitmap bitmap, RectangleF area)

Description:

Add an image to a PDF page at the specified rectangle.

Parameters:

Name	Description	Valid Value
page	The PDF page that will be added an image	Can't be null
bitmap	The image data that will be added to the specified page	Can't be null
area	The rectangle of the added image	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72. The width of the rectangle is larger than 0, less than page width. The height of the rectangle is larger than 0, less

		than page height.

```
public static int AddImage(PDFPage page, Bitmap bitmap, RectangleF area,
AddImageMode mode)
```

Description:

Add an image to a PDF page at the specified rectangle with specified stretch method.

Parameters:

Name	Description	Valid Value
page	The PDF page that will be added an image	Can't be null
bitmap	The image data that will be added to the specified page	Can't be null
area	The rectangle of the added image	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72. The width of the rectangle is larger than 0, less than page width. The height of the rectangle is larger than 0, less than page height.
mode	The stretch method of the added image	FitToArea, FitToWidth, FitToHeight, NoResize

Select Images from PDF page

To select image from PDF page, you can use the following demo codes:

1. Open input PDF file with PDFDocument object.
2. Get the PDFPage object that will be selected image.
3. Call the SelectImage method of PDFImageHandler to select image(s).

The following demo code will show how to select image from PDF page:

C#

```csharp
// Open the document
PDFDocument doc = new PDFDocument(@"C:\input.pdf");
// Get first page of PDF document
PDFPage page = (PDFPage)doc.GetPage(0);
//Set the position
PointF position = new PointF(100F, 200F);
//Select image
PDFImage image = PDFImageHandler.SelectImage(page, position);
```

```vbnet
VB.NET
' Open the document
Dim doc As PDFDocument = new PDFDocument("C:\\input.pdf")
' Get first page of PDF document
Dim page As PDFPage = (PDFPage)doc.GetPage(0)
' Set the position
Dim position As PointF = new PointF(100F, 200F)
' Select image
Dim image As PDFImage = PDFImageHandler.SelectImage(page, position)
```

Related API(s)(**PDFImageHandler.cs**):

public static PDFImage SelectImage(PDFPage page, PointF position)

Description:

> Select the image in the specified position of page.

Parameters:

Name	Description	Valid Value
page	Target page object	Can't be null
position	A position in the page	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72.

Returns:

> PDF Image object if success, null if failed.

public static PDFImage SelectImage(PDFPage page, RectangleF selectedArea)

Description:

> Get the topmost image in the specified region of a page

Parameters:

Name	Description	Valid Value
page	Target page object	Can't be null
selectedArea	A rectangle in the page	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72. The width of the rectangle is larger than 0, less than page width. The height of the rectangle is larger than 0, less than page height.

Returns:

PDF Image object if success, null if failed.

`public static PDFImage SelectImage(PDFPage page, RectangleF selectedArea, int imageIndex)`

Description:

Select the image in the specified region of a page by a sequence index (from bottom to top).

Parameters:

Name	Description	Valid Value
page	Target page object	Can't be null
selectedArea	A rectangle in the page	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72. The width of the rectangle is larger than 0, less than page width. The height of the rectangle is larger than 0, less than page height.
imageIndex	The sequence index, start at 0	Positive int

Returns:

PDF Image object if success, null if imageIndex is out of the sequence or failed.

`public static List<PDFImage> SelectImages(PDFPage page, RectangleF selectedArea)`

Description:

Get all images in the specified region of a page.

Parameters:

Name	Description	Valid Value
page	Target page object	Can't be null
selectedArea	A rectangle in the page	The unit of x,y coordinate is Pixel. The value of x should be 0 to GetWidth() * 72, and the value of y should be 0 to GetHeight() * 72. The width of the rectangle is larger than 0, less than page width. The height of the rectangle is larger than 0, less than page height.

Returns:

A list of image objects if success, null if failed.

Extract Images from PDF File

To extract images from PDF page or file, the following steps will work:

1. Open input PDF file with PDFDocument object.
2. Get the PDFPage object that will be selected image.(Not required)
3. Call the ExtractImages method of PDFImageHandler to extract image(s).

The following codes will show how to extract images from PDF page:

```csharp
C#

// Open the document
PDFDocument doc = new PDFDocument(@"C:\input.pdf");
// Get first page of PDF document
PDFPage page = (PDFPage)doc.GetPage(0);
// Extract all images on one pdf page.
List<PDFImage> allImages = PDFImageHandler.ExtractImages(page);
```

```vbnet
VB.NET

'Open the document
Dim doc As PDFDocument = new PDFDocument("C:\\input.pdf")
'Get first page of PDF document
Dim page As PDFPage = (PDFPage)doc.GetPage(0)
'Extract all images on one pdf page.
Dim allImages = PDFImageHandler.ExtractImages(page)
```

If you want to get more codes, please refer to the sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-image-extract/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-image-extract/

Related API(s)(**PDFImageHandler.cs**):

`public static List<PDFImage> ExtractImages(PDFDocument doc)`

Description:

Extract all images in a PDF document.

Parameters:

Name	Description	Valid Value
doc	Target document object	Can't be null

Returns:

A list of image objects, null if failed.

`public static List<Bitmap> ExtractOriginalImages(PDFDocument doc)`

Description:

Extract all images in a document with their original size.

Parameters:

Name	Description	Valid Value
doc	Target document object	Can't be null

Returns:

A list of image objects, null if failed.


```
public static List<PDFImage> ExtractImages(PDFPage page)
```

Description:

Extract all images in a page.

Parameters:

Name	Description	Valid Value
page	Target page object	Can't be null

Returns:

A list of image objects, null if failed.


```
public static List<Bitmap> ExtractOriginalImages(PDFPage page)
```

Description:

Extract all images in a page with their original size.

Parameters:

Name	Description	Valid Value
page	Target page object	Can't be null

Returns:

A list of image objects, null if failed.

Delete Image from PDF File

To delete image from PDF file:

1. Open input PDF file through PDFDocument object.
2. Call the method ExtractImages of PDFImageHandler to extract images.
3. Call the method DeleteImage to delete image one by one.
4. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

The following code will show you how to remove images from a PDF file:

If you want to know more information, please open the sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-image-remove/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-image-remove/

Related API(s)(**PDFImageHandler.cs**):

```
public static int DeleteImage(PDFDocument doc, PDFImage image)
```

Description:

Delete the specified image from PDF file.

Parameters:

Name	Description	Valid Value
doc	The source PDF file object	Can't be null
image	The image will be deleted	Can't be null

Returns:

Error code, return 0 if success.

Replace Image in PDF page

To replace image in PDF file, you can use the following demo codes:

1. Open input PDF file with PDFDocument object.
2. Get the PDFPage object that will be selected image.
3. Call the SelectImage method of PDFImageHandler to select image(s).
4. Call the ReplaceImage method of PDFImageHandler to replace the old image with new bitmap image.

The following demo code will show how to select image from PDF page:

C#

```csharp
// Open the document
PDFDocument doc = new PDFDocument(@"C:\input.pdf");
// Get first page of PDF document
PDFPage page = (PDFPage)doc.GetPage(0);
//Set the position
PointF position = new PointF(100F, 200F);
//New bitmap
Bitmap bmp = new Bitmap(@"C:\\replace.png");
//Select image
PDFImage image = PDFImageHandler.SelectImage(page, position);
//Replace old image with new bitmap
PDFImageHandler.ReplaceImage(page, image, bmp);
```

VB.NET

```vbnet
' Open the document
Dim doc As PDFDocument = new PDFDocument("C:\\input.pdf")
' Get first page of PDF document
Dim page As PDFPage = (PDFPage)doc.GetPage(0)
' Set the position
Dim position As PointF = new PointF(100F, 200F)
' New bitmap
Dim bmp As Bitmap = new Bitmap(@"C:\\replace.png")
' Select image
Dim image As PDFImage = PDFImageHandler.SelectImage(page, position)
' Replace old image with new bitmap
PDFImageHandler.ReplaceImage(page, image, bmp)
```

Related API(s) (**PDFImageHandler.cs**):

```csharp
public static  int ReplaceImage(PDFPage  page, PDFImage image,    Bitmap
```

```
newImage)
```
Description:

Replace the original image by a new image and keep the location and image size (in page) unchanged.

Parameters:

Name	Description	Valid Value
page	Target page object	Can't be null
image	The image will be replaced	Can't be null
newImage	The new image that is used to replace the old image	Can't be null

Returns:

Error code, return 0 if success.

PDF Conversion

By using RasterEdge.XDoc.PDF, you can convert PDF file to common image formats (.bmp, .jpeg, jpeg2000, .png, .gif and so on), vector image formats (.svg, .html) and some document format (.tif, .docx). While, in order to achieve the conversion (PDF to SVG/HTML), the library RasterEdge.Imaging.SVG.dll will be necessary and RasterEdge.XDoc.Word.dll for PDF converting to Word (.docx).

PDF Convert to Images

To convert PDF file to common format images, you just need do as following steps:

1. Open input PDF file with PDFDocument object.
2. Call the method ConvertToImages of PDFDocument according to your different ways.

The following codes will show how to convert PDF file to jpeg images.

```csharp
C#

// Open the document
PDFDocument doc = new PDFDocument(@"C:\input.pdf");
//Set the output directory
String outputDirectory = @"C:\fileToJpeg\";
// Convert all pages to jpeg images
// It will create jpeg images in the directory whose name is "demo_"
doc.ConvertToImages(ImageType.JPEG, outputDirectory, "demo_");
```

```vbnet
VB.NET

'Open the document
Dim doc As PDFDocument = new PDFDocument("C:\\input.pdf")
'Set the output directory
Dim outputDirectory As String = "C:\\fileToJpeg\\"
'Convert all pages to jpeg images
'It will create jpeg images in the directory whose name is "demo_"
doc.ConvertToImages(ImageType.JPEG, outputDirectory, "demo_")
```

You can choose different image format to convert according to your own need. When running the demo code, please check the output directory must exist on the disk, otherwise it will be failed to convert.

If you want to get more information about converting to images, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-jpeg/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-jpeg/

Related API(s) (*PDFDocument.cs*):

public override void ConvertToImages(ImageType targetType, String directory, String fileName)

Description:

Convert all the PDF pages to target format images and output into the directory.

Parameters:

Name	Description	Valid Value
targetType	The target type image format	The value listed in the class ImageType.cs
directory	The output directory	Must exist on the disk
fileName	The output image file's name without suffix.	Any value whose type is String

public override void ConvertToImages(ImageType targetType, Stream[] streams)

Description:

Convert all the PDF pages to target format images and save them to streams

Parameters:

Name	Description	Valid Value
targetType	The target type image format	The value listed in the class ImageType.cs
streams	The output streams	Any valid stream like FileStream or MemoryStream.

```
public override void ConvertToImages(ImageType targetType, float zoomValue,
String directory, String fileName)
```
Description:

Convert all the PDF pages to target format images with zoomValue, and output into the directory.

Parameters:

Name	Description	Valid Value
targetType	The target type image format	The value listed in the class ImageType.cs
zoomValue	The magnification of the original PDF page size	0.1 ~ 20f
directory	The output directory	Must exist on the disk
fileName	The output image file's name without suffix.	Any value whose type is String

```
public override void ConvertToImages(ImageType targetType, float zoomValue,
Stream[] streams)
```
Description:

Convert all the PDF pages to target format images with specified zoom and save them to streams

Parameters:

Name	Description	Valid Value
targetType	The target type image format	The value listed in the class ImageType.cs
zoomValue	The magnification of the original PDF page size	0.1 ~ 20f
streams	The output streams	Any valid stream like FileStream or MemoryStream.

```
public override void ConvertToImages(ImageType targetType, int resolution,
String directory, String fileName)
```
Description:

Convert all the PDF pages to target format images with specified resolution and output into the directory.

Parameters:

Name	Description	Valid Value
targetType	The target type image format	The value listed in the class ImageType.cs
resolution	The output image resolution	Integer
directory	The output directory	Must exist on the disk
fileName	The output image file's name without suffix.	Any value whose type is String

```
public override void ConvertToImages(ImageType targetType, int resolution,
Stream[] streams)
```
Description:

 Convert all the PDF pages to target format images with specified resolution and save them to streams

Parameters:

Name	Description	Valid Value
targetType	The target type image format	The value listed in the class ImageType.cs
resolution	The output image resolution	Integer
streams	The output streams	Any valid stream like FileStream or MemoryStream.

```
public    override    void    ConvertToImages(ImageType    targetType,
ImageOutputOption option, String directory, String fileName)
```
Description:

 Convert all the PDF pages to target format images with settings through option and output into the directory.

Parameters:

Name	Description	Valid Value
targetType	The target type image format	The value listed in the class ImageType.cs
option	The settings of the output images	-
directory	The output directory	Must exist on the disk
fileName	The output image file's name without suffix.	Any value whose type is String

```
public    override    void    ConvertToImages(ImageType    targetType,
ImageOutputOption option, Stream[] streams)
```
Description:

 Convert all the PDF pages to target format images with settings through option and save them to streams

Parameters:

Name	Description	Valid Value
targetType	The target type image format	The value listed in the class ImageType.cs
option	The settings of the output images	-
streams	The output streams	Any valid stream like FileStream or MemoryStream.

PDF Convert to Vector Images

As SVG and HTML images are defined in XML text lines, they can be easily searched, indexed, scripted, and supported by most of the up to date web browsers. Therefore, in C#.NET web document viewing applications, PDF is often rendered and converted to SVG image for high fidelity viewing. With RasterEdge.XDoc.PDF, you can convert PDF file to SVG/HTML images easily.

Add extra references:

    If you want to convert PDF to SVG/HTML Image, you need add the following dll:

- RasterEdge.Imaging.SVG.dll

To convert PDF file to SVG/HTML images, following steps is enough:

1. Open input PDF file with PDFDocument object.
2. Call the method ConvertToVectorImages.

The following demo code will tell you how to convert PDF file to SVG files:

C#

```csharp
// Open the document
PDFDocument doc = new PDFDocument(@"C:\input.pdf");
//Set the output directory
String outputDirectory = @"C:\fileToSVG\";
// Convert all pages to SVG images
// It will create SVG files in the directory whose name is "demo_"
doc.ConvertToVectorImages(ContextType.SVG, outputDirectory, "demo_",
RelativeType.SVG);
```

VB.NET

```vbnet
'Open the document
Dim doc As PDFDocument = new PDFDocument("C:\\input.pdf")
'Set the output directory
Dim outputDirectory As String = "C:\\fileToSVG\\"
'Convert all pages to SVG images
'It will create SVG files in the directory whose name is "demo_"
doc.ConvertToVectorImages(ContextType.SVG, outputDirectory, "demo_",
RelativeType.SVG)
```

Also, you can refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-svg/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-svg/
http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-html/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-html/

Related API(s) (**PDFDocument.cs**):

```
public override void ConvertToVectorImages(ContextType targetType, String
directory, String fileName, RelativeType type)
```

Description:

Convert all the PDF pages to SVG files and save into the directory.

Parameters:

Name	Description	Valid Value
targetType	The target type image format	ContextType.SVG ContextType.HTML
directory	The output directory to save the svg files	Must exist on the disk
fileName	The output svg file name	Any value whose type is String
type	The method to write the output font	When on the IIS, the value is RelativeType.ASP, RelativeType.SVG/RelativeType.HTML if it is running on the Visual Stuido.

PDF Convert to Document

Add extra references:

If you want to convert PDF to Tiff Image, you need add the following dll:

- RasterEdge.XDoc.TIFF.dll

If you want to convert PDF to Word(.docx) file, you need add the following dlls:

- RasterEdge.XDoc.Office.Inner.Common.dll
- RasterEdge.XDoc.Office.Inner.Office03.dll
- RasterEdge.XDoc.Word.dll

By using RasterEdge.XDoc.PDF, you can convert PDF file to tiff and word (.docx) file.

To achieve the conversion, please do as follows:

1. Open input PDF file through PDFDocument object.
2. Call the method ConvertToDocument to complete the conversion.

The following demo code will explain how to convert PDF file to TIFF/Word:

```csharp
C#

// Open the document
PDFDocument doc = new PDFDocument(@"C:\input.pdf");
//Set the output file path
String outputFilePath = @"C:\output.tif";
// Convert PDF file to tiff file
doc.ConvertToDocument(DocumentType.SVG, outputFilePath);
```

Also, you can refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-tiff/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-tiff/
http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-word/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-word/

Related API(s) (**PDFDocument.cs**):

```csharp
public override void ConvertToDocument(DocumentType targetType, String filePath)
```

Description:

Convert PDF file to TIFF/DOCX and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
filePath	The output file path	-

```csharp
public override void ConvertToDocument(DocumentType targetType, Stream stream)
```

Description:

Convert PDF file to TIFF/DOCX and save it into stream.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
stream	The output stream.	Any valid FileStream or MemoryStream.

```csharp
public override void ConvertToDocument(DocumentType targetType, float zoomValue, String filePath)
```

Description:

Convert PDF file to TIFF/DOCX with specified zoom value and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The target document type will	DocumentType.TIFF

	be converted.	DocumentType.DOCX
zoomValue	The magnification of the original PDF page size	0.1～20F
filePath	The output file path	-

`public override void ConvertToDocument(``DocumentType`` targetType, ``float``
zoomValue, `<span style="color:teal">`Stream`</span>` desStream)`

Description:

Convert PDF file to TIFF/DOCX with specified zoom value and save it into stream.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
zoomValue	The magnification of the original PDF page size	0.1～20F
desStream	The output stream.	Any valid FileStream or MemoryStream.

`public override void ConvertToDocument(``DocumentType`` targetType, ``int``
resolution, `<span style="color:blue">`string`</span>` filePath)`

Description:

Convert PDF file to TIFF/DOCX with specified resolution and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
resolution	The target resolution of the output tiff file, it is invalid for word file.	Integer
filePath	The output file path	-

`public override void ConvertToDocument(``DocumentType`` targetType, ``int``
resolution, `<span style="color:teal">`Stream`</span>` desStream)`

Description:

Convert PDF file to TIFF/DOCX with specified resolution and save it into stream.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
resolution	The target resolution of the output tiff file, it is invalid for word file.	Integer

| desStream | The output stream. | Any valid FileStream or MemoryStream. |

```
public    override    void    ConvertToDocument(DocumentType    targetType,
ImageCompress compression, String filePath)
```
Description:
    Convert PDF file to TIFF/DOCX with specified compression method and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
compression	The target compression of the output tiff file, it is invalid for word file.	The type listed in the ImageCompress.cs
filePath	The output file path	-

```
public    override    void    ConvertToDocument(DocumentType    targetType,
ImageCompress compression, Stream desStream)
```
Description:
    Convert PDF file to TIFF/DOCX with specified compression method and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
compression	The target compression of the output tiff file, it is invalid for word file.	The type listed in the ImageCompress.cs
desStream	The output stream	Any valid FileStream or MemoryStream.

```
public override void ConvertToDocument(DocumentType targetType, string
filePath, ImageOutputOption options)
```
Description:
    Convert PDF file to TIFF/DOCX with specified settings through options method and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
filePath	The output file path	-
options	The settings of the output tiff file, it is invalid for word file.	-

```
public override void ConvertToDocument(DocumentType targetType,    Stream
```

```
desStream, ImageOutputOption options)
```
Description:

Convert PDF file to TIFF/DOCX with specified settings through options method and save it into stream.

Parameters:

Name	Description	Valid Value
targetType	The target document type will be converted.	DocumentType.TIFF DocumentType.DOCX
desStream	The output stream	Any valid FileStream or MemoryStream.
options	The settings of the output tiff file, it is invalid for word file.	-

PDF Convert to Text

Please refer to the following site:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-text/

PDF Generator

In order to create PDF from different document, our RasterEdge.XDoc.PDF allow developer to generator PDF from PDF, Office, Open Office, CSV, RTF, Text, Image and Create empty PDF file.

Extract PDF document from PDF file

To extract a PDF file from source PDF file, there are two ways to complete the task:
First way:

1. Open input PDF file through PDFDocument object.
2. Call the GetMultiDocument to extract a new PDFDocument object.
3. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

The following demo code will show how to extract new PDF file:

C#

```csharp
// Open the document
PDFDocument doc = new PDFDocument(@"C:\input.pdf");
//Set the output file path
String outputFilePath = @"C:\output.pdf";
//The page index will be extract of the pdf file, starting at 0
//The first, fouth, fifth pages will be extracted.
List<int> pageIndexes = new List<int>();
pageIndexes.Add(0);
pageIndexes.Add(3);
pageIndexes.Add(5);
// Extract new PDFDocument object
PDFDocument newDocument = doc.GetMultiDocument(pageIndexes);
// Save the new PDFDocument on the disk.
doc.Save(outputFilePath);
```

```vbnet
'Open the document
Dim doc As PDFDocument = new PDFDocument("C:\\input.pdf")
'Set the output file path
Dim outputFilePath As String = " C:\\output.pdf "
'The page index will be extract of the pdf file, starting at 0
'The first, fouth, fifth pages will be extracted.
Dim pageIndexes As List<Of Integer> = new List<Of Ingeter>()
pageIndexes.Add(0)
pageIndexes.Add(3)
pageIndexes.Add(5)
// Extract new PDFDocument object
Dim newDocument As PDFDocument = doc.GetMultiDocument(pageIndexes)
// Save the new PDFDocument on the disk.
doc.Save(outputFilePath)
```

Second way:

1. Call the static method ExtractDocument to extract a new PDFDocument directly.

C#

```csharp
//Set the output file path
String outputFilePath = @"C:\output.pdf";
//The page index will be extract of the pdf file, starting at 0
//The first, fouth, fifth pages will be extracted.
List<int> pageIndexes = new List<int>();
pageIndexes.Add(0);
pageIndexes.Add(3);
pageIndexes.Add(5);
// Extract new PDFDocument object
PDFDocument.ExtractDocument(@"C:\input.pdf", pageIndexes.ToArray(),
outputFilePath);
```

Related API(s) (**PDFDocument.cs**):

`public static int ExtractDocument(String filePath, int[] pageIndexes)`

Description:

Extract a new PDF file with specified pages and cover the old PDF file.

Parameters:

Name	Description	Valid Value
filePath	The input PDF file.	An existing PDF file path.
pageIndexes	The page indexes will be extracted which is starting at 0.	The page index is 0 ~ page count - 1

`public static int ExtractDocument(String inputFilePath, int[] pageIndexes, String outputFilePath)`

Description:

Extract a new PDF file with specified pages and save it to new output file path.

Parameters:

Name	Description	Valid Value
filePath	The input PDF file.	An existing PDF file path.
pageIndexes	The page indexes will be extracted which is starting at 0.	The page index is 0 ~ page count - 1
outputFilePath	The new output file path.	

`public BaseDocument GetMultiDocument(List<int> pageIdxes)`

Description:

Extract a new PDFDocument object from the input PDFDocument with specified page indexes.

Parameters:

Name	Description	Valid Value
pageIndexes	The page indexes will be extracted which is starting at 0.	The page index is 0 ~ page count - 1

Return:

A new PDFDocument object if success, null if failed.

Create empty file

When you need create a new PDF file, then add text, image, bookmark and so on into the new PDF file, it will be very useful for you to use RasterEdge.XDoc.PDF library.

To create a new PDF file, you just need to call the static method Create directly, then you will get a new PDFDocument object to do more operators.

C#

```csharp
//Set the output file path
String outputFilePath = @"C:\output.pdf";
// Create new PDFDocument object
// It will return a new PDFDocument object including two empty pages.
// The page size will be default A4.
PDFDocument doc = PDFDocument.Create(2);
// Save the new PDF on the disk
doc.Save(outputFilePath);
```

VB.NET

```vbnet
'Set the output file path
Dim outputFilePath As String = " C:\\output.pdf "
'Create new PDFDocument object
'It will return a new PDFDocument object including two empty pages.
'The page size will be default A4.
Dim doc As PDFDocument = PDFDocument.Create(2)
'Save the new PDF on the disk.
doc.Save(outputFilePath)
```

Related API(s) (**PDFDocument.cs**):

```
public static PDFDocument Create(int pageCount)
```
Description:

Create a new empty PDF file, the page count is pageCount and the size of every page is A4.

Parameters:

Name	Description	Valid Value
pageCount	The page count of new PDF	The minimum is 1

Return:

A new PDFDocument object, null if failed.

```
public static PDFDocument Create(int pageCount, PaperSize paperSize)
```

Description:

Create a new empty PDF file, the total page count is set by pageCount and the size of every page is set by paperSize.

Parameters:

Name	Description	Valid Value
pageCount	The page count of new PDF	The minimum is 1
paperSize	The size of new PDF page	The value listed in the PaperSize.cs

Return:

A new PDFDocument object, null if failed.

public static PDFDocument Create(int pageCount, float width, float height)

Description:

Create a new empty PDF file with specified page width and height, the total page count is set by pageCount.

Parameters:

Name	Description	Valid Value
pageCount	The page count of new PDF	The minimum is 1
width	The width of new PDF page	The unit is Pixel. The value should larger than 0.
height	The height of new PDF page	The unit is Pixel. The value should larger than 0.

Return:

A new PDFDocument object, null if failed.

public static PDFDocument Create(int pageCount, List<RectangleF> rectangles)

Description:

Create a new empty PDF file with specified size which is same to the size of rectangles, the total page count is set by pageCount.

Parameters:

Name	Description	Valid Value
pageCount	The page count of new PDF	The minimum is 1
rectangles	The source size will be used as the size of new PDF pages.	The unit of the member size should be Pixel. The value should not be null.

Return:

A new PDFDocument object, null if failed.

public static PDFDocument Create(Bitmap image, ImageToPDFSetting buildSetting)

Description:

Create a new empty PDF file from source image with settings through buildSetting.

Parameters:

Name	Description	Valid Value
image	The image used to create new PDF file	Can't be null.
buildSetting	The new PDF compression and more settings when image convert to PDF file.	The value should not be null.

Return:

A new PDFDocument object, null if failed.

```
public static PDFDocument Create(Bitmap[] images, ImageToPDFSetting
buildSetting)
```

Description:

Create a new empty PDF file from source images with settings through buildSetting. The page count will be equal to the number of images.

Parameters:

Name	Description	Valid Value
images	Source images used to create PDF file.	Every image should not be null
buildSetting	The new PDF compression and more settings when image convert to PDF file.	The value should not be null.

Return:

A new PDFDocument object, null if failed.

Create PDF file from Office

Please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-word-to-pdf/

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-excel-to-pdf/

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-powerpoint-to-pdf/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-word-to-pdf/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-excel-to-pdf/

http://www.rasteredge.com/how-to/ vb-net -imaging/pdf-convert-powerpoint-to-pdf/

Create PDF file from Open Office

Please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-odt-to-pdf/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-odt-to-pdf/

Create PDF from Tiff

Please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-tiff-to-pdf/
http://www.rasteredge.com/how-to/ vb-net-imaging/pdf-convert-tiff-to-pdf/

Create PDF from CSV

Please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-csv-to-pdf/
http://www.rasteredge.com/how-to/ vb-net-imaging/pdf-convert-csv-to-pdf/

Create PDF from RTF

Please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-rtf-to-pdf/
http://www.rasteredge.com/how-to/ vb-net-imaging/pdf-convert-rtf-to-pdf/

Create PDF from Text

Please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-text-to-pdf/
http://www.rasteredge.com/how-to/ vb-net-imaging/pdf-convert-text-to-pdf/

Working with Page

Get page Properties

To get the properties of PDF page, you can do as following steps:

1. Open an input PDF file with PDFDocument object.
2. Get the PDFPage object by calling the method ***GetPage***.
3. Call the corresponding method to get the property.

The following demo codes will show you how to do:

C#

```csharp
'Open a document.
String inputFilePath = "C:\\1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get the first page from the document.
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);
//Get page width whose unit is inch.
float width = page.GetWidth();
//Get page height whose unit is inch.
float height = page.GetHeight();
//Get page rotation degree whose value is 90,180,270.
float rotateDegree = page.GetRotation();
```

VB.NET

```vbnet
'Open a document.
Dim inputFilePath As String = "C:\\1.pdf"
Dim doc As PDFDocument = new PDFDocument(inputFilePath)
// Get the first page from the document.
Dim pageIndex As Integer = 0;
Dim page As PDFPage = (PDFPage)doc.GetPage(pageIndex)
//Get page width whose unit is inch.
Dim width = page.GetWidth()
//Get page height whose unit is inch.
Dim height = page.GetHeight()
//Get page rotation degree whose value is 90,180,270.
Dim rotateDegree = page.GetRotation()
```

Related API(s) (**PDFPage.cs**):

```
public override float GetHeight()
```
Description:

   Get the PDF page height, and the unit is *Inch*.

Return:

   It is the value of the PDF page height whose unit is *Inch*.


```
public override float GetWidth()
```
Description:

   Get the PDF page width, and the unit is *Inch*.

Return:

   It is the value of the PDF page width whose unit is *Inch*.


```
public override float GetRotation()
```
Description:

   Get the PDF page rotate degree, and the value is *90,180 or 270*.

Return:

   It is the value of the PDF page rotation.

Convert to Image

To convert one PDF page to image, you just need several steps as follow:

1. Open an existing PDF file through PDFDocument object.
2. Get the PDFPage object that will be converted to image through calling the method GetPage.
3. Call the method ConvertToImage, ConvertToImageBytes, ConvertToImageStream, ConvertToImageFitHeight and ConvertToImageWidth to convert the PDF page to image and save it to file/Stream/Byte Array.

The following demo code will show the conversion in details:

C#

```
'Open a document.
String inputFilePath = "C:\\input.pdf";
String outputImageFilePath = "C:\\output.png";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get the first page from the document.
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);
//Convert PDF page to image
Bitmap bmp = page.ConvertToImage();
//Save the bitmap to a file.
bmp.Save(outputImageFilePath);
```

VB.NET

```
'Open a document.
Dim inputFilePath As String = "C:\\input.pdf"
Dim outputImageFilePath As String = "C:\\output.png"
Dim doc As PDFDocument = new PDFDocument(inputFilePath)
// Get the first page from the document.
Dim pageIndex As Integer = 0
Dim page As PDFPage = (PDFPage)doc.GetPage(pageIndex)
'Convert PDF page to image
Dim bmp As Bitmap = page.ConvertToImage()
'Save the bitmap to a file.
bmp.Save(outputImageFilePath)
```

If you want to get more demo codes, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-convert-raster/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-convert-raster/

Related API(s) (**PDFPage.cs**):

```
public override Bitmap ConvertToImage()
```
Description:

    Convert the PDF page to bitmap as the default settings, source page size, resolution = 96.

Return:

    A bitmap object, null if failed.

```
public override Bitmap ConvertToImage(float zoomValue)
```
Description:

    Convert the PDF page to bitmap with specified zoom value.

Parameters:

Name	Description	Valid Value
zoomValue	The magnification of the output image.	0.1 ~ 20f

Return:

    A bitmap object, null if failed.

```
public override Bitmap ConvertToImage(int targetResolution)
```
Description:

    Convert the PDF page to bitmap with specified resolution.

Parameters:

Name	Description	Valid Value
resolution	The resolution of the output image.	Integer, larger than 0.

Return:

    A bitmap object, null if failed.

```
public override Bitmap ConvertToImage(Size targetSize)
```
Description:

    Convert the PDF page to bitmap with specified image size.

Parameters:

Name	Description	Valid Value
targetSize	The size of the output image.	Can't be null, and the width and height must be larger than 0.

Return:

    A bitmap object, null if failed.

```
public override Bitmap ConvertToImageFitHeight(int height)
```
Description:

    Convert the PDF page to bitmap with specified height and the width will be zoomed as scaling height.

Parameters:

Name	Description	Valid Value

height	The height of the output image.	Must be larger than 0.

Return:

A bitmap object, null if failed.

public override Bitmap ConvertToImageFitWidth(int width)

Description:

Convert the PDF page to bitmap with specified width and the height will be zoomed as scaling width.

Parameters:

Name	Description	Valid Value
width	The width of the output image.	Must be larger than 0.

Return:

A bitmap object, null if failed.

public override void ConvertToImage(ImageType targetType, String filePath)

Description:

Convert the PDF page to bitmap with specified format and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The format of the output image byte array.	The value must be one listed in the ImageType.cs
filePath	The output image file path	An valid file path

public override void ConvertToImage(ImageType targetType, float zoomValue, String filePath)

Description:

Convert the PDF page to bitmap with specified format and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The format of the output image byte array.	The value must be one listed in the ImageType.cs
zoomValue	The zoom value of output image	0.1 ~ 20f
filePath	The output image file path	An valid file path

public override void ConvertToImage(ImageType targetType, int resolution, String filePath)

Description:

Convert the PDF page to bitmap with specified format and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The format of the output	The value must be one listed

	image byte array.	in the ImageType.cs
resolution	The resolution of output image	Integer
filePath	The output image file path	An valid file path

public override byte[] ConvertToImageBytes(ImageType targetType)

Description:

Convert the PDF page to bitmap byte array with specified image format.

Parameters:

Name	Description	Valid Value
targetType	The format of the output image byte array.	The value must be one listed in the ImageType.cs

Return:

A byte array, null or empty byte array if failed.

public override byte[] ConvertToImageBytes(ImageType targetType, float zoomValue)

Description:

Convert the PDF page to bitmap byte array with specified image format and the size will be zoomed as specified by zoomValue.

Parameters:

Name	Description	Valid Value
targetType	The format of the output image byte array.	The value must be one listed in the ImageType.cs
zoomValue	The zoom value of the output image size	0.1 ~ 20f

Return:

A byte array, null or empty byte array if failed.

public override byte[] ConvertToImageBytes(ImageType targetType, int targetResolution)

Description:

Convert the PDF page to bitmap byte array with specified image format and resolution

Parameters:

Name	Description	Valid Value
targetType	The format of the output image byte array.	The value must be one listed in the ImageType.cs
resolution	The resolution of the output image size	Integer

Return:

A byte array, null or empty byte array if failed.

public override void ConvertToImageStream(ImageType targetType, Stream stream)

Description:

      Convert the PDF page to bitmap with specified format and save it into one stream.

Parameters:

Name	Description	Valid Value
targetType	The format of the output image byte array.	The value must be one listed in the ImageType.cs
stream	The stream to save the output image data	Can't be null

public override void ConvertToImageStream(ImageType targetType, float zoomValue, Stream stream)

Description:

      Convert the PDF page to bitmap with specified format, zoom value and save it into one stream.

Parameters:

Name	Description	Valid Value
targetType	The format of the output image byte array.	The value must be one listed in the ImageType.cs
zoomValue	The zoom value of the output image	0.1 ~ 20f
stream	The stream to save the output image data	Can't be null

public override void ConvertToImageStream(ImageType targetType, int resolution, Stream stream)

Description:

      Convert the PDF page to bitmap with specified format, resolution and save it into one stream.

Parameters:

Name	Description	Valid Value
targetType	The format of the output image byte array.	The value must be one listed in the ImageType.cs
resolution	The resolution of the output image	Integer
stream	The stream to save the output image data	Can't be null

public override void ConvertToImage(ImageType targetType, ImageOutputOption option, String filePath)

Description:

      Convert the PDF page to bitmap with specified settings by option and save it on the disk.

Parameters:

Name	Description	Valid Value
targetType	The format of the output	The value must be one listed

	image byte array.	in the ImageType.cs
`option`	The settings of output image	Can't be null
`filePath`	The output image file path	Valid file path

`public` `override` `void` `ConvertToImageStream(`ImageType` targetType, `ImageOutputOption` option, `Stream` stream)`

Description:

Convert the PDF page to bitmap with specified settings by option and save it into one stream.

Parameters:

Name	Description	Valid Value
`targetType`	The format of the output image byte array.	The value must be one listed in the ImageType.cs
`option`	The settings of output image	Can't be null
`stream`	The stream to save the output image data	Can't be null

Convert to Vector Image

To convert one PDF page to vector image(SVG,HTML), the following necessary is enough:

1. Open an existing PDF file through PDFDocument object.
2. Get the PDFPage object by calling the method GetPage.
3. Call the method ConvertToVectorImage to achieve the conversion.

The following demo code will show you how to complete the conversion:

C#

```csharp
'Open a document.
String inputFilePath = "C:\\input.pdf";
String outputSvgDirectory = "C:\\svgOutput\\";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get the first page from the document.
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);
//Convert PDF page to svg file
page.ConvertToVectorImage(ContextType.SVG, outputSvgDirectory, "demo_",
RelativeType.SVG);
```

VB.NET

```vbnet
'Open a document.
Dim inputFilePath As String = "C:\\input.pdf"
Dim outputSvgDirectory As String = " C:\\svgOutput\\"
Dim doc As PDFDocument = new PDFDocument(inputFilePath)
// Get the first page from the document.
Dim pageIndex As Integer = 0
Dim page As PDFPage = (PDFPage)doc.GetPage(pageIndex)
'Convert PDF page to svg file
page.ConvertToVectorImage(ContextType.SVG, outputSvgDirectory, "demo_",
RelativeType.SVG)
```

Related API(s) (**PDFPage.cs**):

```
public override void ConvertToVectorImage(ContextType targetType, String
directory, String fileName, RelativeType type)
```

Description:

　　Convert all the PDF pages to SVG files and save into the directory.

Parameters:

Name	Description	Valid Value
targetType	The target type image format	ContextType.SVG ContextType.HTML
directory	The output directory to save the svg files	Must exist on the disk
fileName	The output svg file name	Any value whose type is String
type	The method to write the output font	When on the IIS, the value is RelativeType.ASP, RelativeType.SVG/RelativeType.HTML if it is running on the Visual Stuido.

Working with Header/Footer

By using XDoc.PDF library, the region of header/footer is defined through Page Margins.
Header: The area surrounded by Left Margin, Right Margin, Top Margin.
Footer: The area surrounded by Left Margin, Right Margin, Bottom Margin.

The header/footer region can be divided into 3 PDFPageTextField (Left/Center/Right).
You can define text font and color trough PDFPageTextField object.
The region and location of header/footer are as follows:
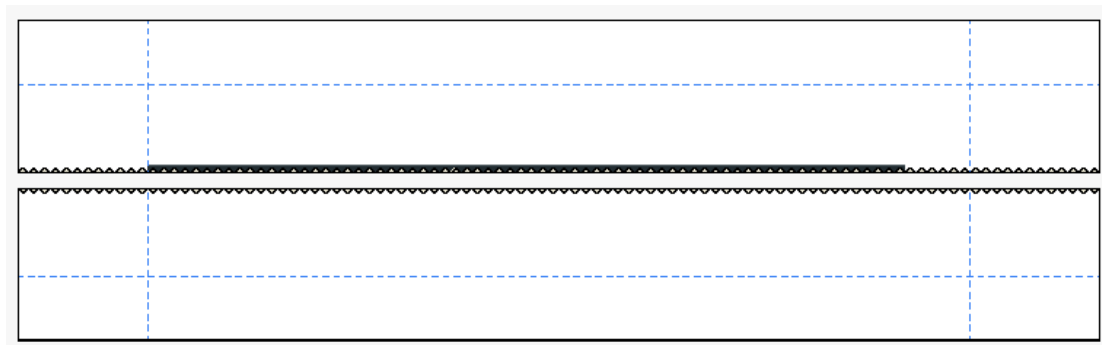
The following demo codes will show you how to set margins to create Header/Footer regions.

```
C#

//Create a Header/Footer object.
PDFPageHeaderFooter hdrftr = new PDFPageHeaderFooter();
//Define Marings to create a Header/Footerregion, Unit: inches
hdrftr.LeftMargin = 0.5F;
hdrftr.RightMargin = 0.5F;
hdrftr.TopMargin = 1.0F;
hdrftr.BottomMargin = 1.0F;


//Set the properties of Text Fields
//Set LeftHeader Field
hdrftr.LeftHeaderField.TextFont = new Font("Arial", 12F, FontStyle.Regular);
hdrftr.LeftHeaderField.TextColor = Color.Black;
hdrftr.LeftHeaderField.SetContent("This is a Header field");


//Set other Fields
//hdrftr.CenterHeaderField…
//hdrftr.RightHeaderField…
//hdrftr.LeftFooterField…
//hdrftr.CenterFooterField…
//hdrftr.RightFooterField…
```

```
VB.NET

'Create a Header/Footer object.
Dim hdrftr As PDFPageHeaderFooter = New PDFPageHeaderFooter();
'Define Marings to create a Header/Footerregion, Unit: inches
hdrftr.LeftMargin = 0.5F;
hdrftr.RightMargin = 0.5F;
hdrftr.TopMargin = 1.0F;
hdrftr.BottomMargin = 1.0F;

//Set the properties of Text Fields
//Set LeftHeader Field
hdrftr.LeftHeaderField.TextFont = New Font("Arial", 12F, FontStyle.Regular);
hdrftr.LeftHeaderField.TextColor = Color.Black;
hdrftr.LeftHeaderField.SetContent("This is a Header field");

'Set other Fields
'hdrftr.CenterHeaderField…
'hdrftr.RightHeaderField…
'hdrftr.LeftFooterField…
'hdrftr.CenterFooterField…
'hdrftr.RightFooterField…
```

If you want to add header/footer to some pages not all pages, you can complete the task through the PageRangeOptions object.

All Pages:
```
PageRangeOptions ops = new PageRangeOptions();
ops.AllPages = true;
```

From Third to Sixth page:
```
PageRangeOptions ops = new PageRangeOptions();
ops.SetPageNumberRange(3, 6);
```

All the Even pages from Third to Sixth pages(That is 4th,6th page):
```
PageRangeOptions ops = new PageRangeOptions();
ops.SetPageNumberRange(3, 6);
ops.Subset = PageRangeSubset.Even;
```

Get page index array or page number array:
```
PageRangeOptions ops = new PageRangeOptions();
//  ...

int totalPageCount = 20;
```

```
int[]    pageIndexes    =    PageRangeOptions.CalculatePageIndexes(ops,
totalPageCount);
int[] pageNumbers = PageRangeOptions.CalculatePageNumbers(ops,
totalPageCount);
```

Note:
a. When adding Header/Footer to page, it will cover current header/footer settings, the document will record the value set last time.
b. When deleting Header/Footer, it will delete total header/footer, you can't delete header/footer partially.
c. When adding Header/Footer to page, it will be better to delete the recorded settings, otherwise the document will apply new and old settings at the same time.

Add Header/Footer to Document

The following demo codes will show how to add header/footer to document:

```csharp
PDFDocument doc = new PDFDocument(@"C:\Content.pdf");
//Define a header/footer setting
PDFPageHeaderFooter headerFooter = new PDFPageHeaderFooter();
//Set center header field
headerFooter.CenterHeaderField.Set(@"Title:*****", new Font("Arial", 12F,
FontStyle.Regular), Color.Black, false);
//Set left footer field
headerFooter.LeftFooterField.Set("Page<<1/n>>", new Font("Arial", 9F,
FontStyle.Regular), Color.DarkGray, false);


//Define page range:all odd pages
PageRangeOptions pageRange = new PageRangeOptions();
pageRange.AllPages = true;
pageRange.Subset = PageRangeSubset.Odd;


//Apply header/footer settings to all odd pages
PDFPageFieldHandler.ApplyHeaderFooter(doc, headerFooter, pageRange);


//  define a header/footer setting
PDFPageHeaderFooter hdrftr2 = new PDFPageHeaderFooter();
//  set center header field
hdrftr2.CenterHeaderField.Set(@"Title: *****", new Font("Arial", 12F,
FontStyle.Regular), Color.Black, false);
//  set right footer field
hdrftr2.RightFooterField.Set("Page <<1/n>>", new Font("Arial", 9F,
FontStyle.Regular), Color.DarkGray, false);


//  define page range: all even pages
PageRangeOptions pageRange2 = new PageRangeOptions();
pageRange2.AllPages = true;
pageRange2.Subset = PageRangeSubset.Even;


// apply header/footer settings to all even pages
PDFPageFieldHandler.ApplyHeaderFooter(doc, hdrftr2, pageRange2);
doc.Save(@"C:\output.pdf");
```

```vbnet
VB.NET

Dim doc As PDFDocument = New PDFDocument("C:\\input.pdf")
'Define a header/footer setting
Dim headerFooter As PDFPageHeaderFooter = New PDFPageHeaderFooter()
'Set center header field
headerFooter.CenterHeaderField.Set("Title:*****", New Font("Arial", 12.0F,
FontStyle.Regular), Color.Black, False)
headerFooter.CenterHeaderField.Set("Title:*****", New Font("Arial", 12.0F,
FontStyle.Regular), Color.Black)
'Set left footer field
headerFooter.LeftFooterField.Set("Page<<1/n>>", New Font("Arial", 9.0F,
FontStyle.Regular), Color.DarkGray, False)
'Define page range:all odd pages
Dim pageRange As PageRangeOptions = New PageRangeOptions()
pageRange.AllPages = True
pageRange.Subset = PageRangeSubset.Odd


'Apply header/footer settings to all odd pages
PDFPageFieldHandler.ApplyHeaderFooter(doc, headerFooter, pageRange)


'Define a header/footer setting
Dim hdrftr2 As PDFPageHeaderFooter = New PDFPageHeaderFooter()
'Set center header field
hdrftr2.CenterHeaderField.Set("Title: *****", New Font("Arial", 12.0F,
FontStyle.Regular), Color.Black, False)
'Set right footer field
hdrftr2.RightFooterField.Set("Page <<1/n>>", New Font("Arial", 9.0F,
FontStyle.Regular), Color.DarkGray, False)


'Define page range: all even pages
Dim pageRange2 As PageRangeOptions = New PageRangeOptions()
pageRange2.AllPages = True
pageRange2.Subset = PageRangeSubset.Even
'Apply header/footer settings to all even pages
PDFPageFieldHandler.ApplyHeaderFooter(doc, hdrftr2, pageRange2)
doc.Save("C:\\output.pdf")
```

Related API(s) (**PDFPageFieldHandler.cs**):

`public static int` ApplyHeaderFooter(`PDFDocument` doc, `PDFPageHeaderFooter`
hdrftr, `PageRangeOptions` ops);

Description:

Add header/footer to PDF file with specified settings.

Parameters:

Name	Description	Valid Value
doc	Input pdf file	Can't be null
hdrftr	Header/Footer object	Can't be null
ops	Specific settings	-

Return:

Error code, 0 if success.

Remove Header/Footer from document

The following demo code will explain how to delete header and footer from pdf document.

```csharp
C#

PDFDocument doc = new PDFDocument(inputFilePath);
PDFPageFieldHandler.RemoveHeaderFooters(doc);
doc.Save(outputFilePath);
```

```vbnet
VB.NET

Dim doc As PDFDocument = New PDFDocument(inputFilePath)
PDFPageFieldHandler.RemoveHeaderFooters(doc)
doc.Save(outputFilePath)
```

Related API(s) (**PDFPageFieldHandler.cs**):

```
public static int RemoveHeaderFooters(PDFDocument doc);
```

Description:

Remove header/footer from pdf document.

Parameters:

Name	Description	Valid Value
doc	Input pdf file	Can't be null

Return:

Error code, 0 if success.

```
public static int RemoveHeaderFooters(PDFDocument doc, int[] pageIndexs);
```

Description:

Remove page header/footer from specified pdf pages.

Parameters:

Name	Description	Valid Value
doc	Input pdf file	Can't be null
pageIndexs	An array of page index of pages that will be deleted header/footer	0 ~ page number – 1, which is starting at 0.

Return:

Error code, 0 if success.

Working with Watermark

Watermark Resource

With XDoc.PDF library, you can create a watermark with following resources:

- Text
- Image
- PDF File

The following demo code will show you how to create watermark with Textsource.

```csharp
C#

String textContent = "Confidential";
Font textFont = new Font("Arial", 24F, FontStyle.Regular);
Color textColor = Color.Black;
WatermarkTextAlignment textLineAlignment = WatermarkTextAlignment.Center;
PDFWatermarkRes resWatermark = new PDFWatermarkTextRes(textContent,
textFont, textColor, textLineAlignment);
```

```vbnet
VB.NET

Dim textContent As String = "Confidential"
Dim textFont As Font = New Font("Arial", 24.0F, FontStyle.Regular)
Dim textColor As Color = Color.Black
Dim textAlignment As WatermarkTextAlignment = WatermarkTextAlignment.Center
Dim resWatermark As PDFWatermarkRes = New PDFWatermarkTextRes(textContent,
textFont, textColor, textLineAlignment)
```

The following demo code will explain how to create image watermark with XDoc.PDF.

```csharp
C#

Bitmap image = new Bitmap("C:\\logo.jpg");
PDFWatermarkRes resWatermark = new PDFWatermarkImageRes(image);
```

```vbnet
VB.NET

Dim image As Bitmap = New Bitmap("C:\\logo.jpg")
Dim resWatermark As PDFWatermarkRes = New PDFWatermarkImageRes(image)
```

The following demo code will allow you to create a watermark with PDF File source.

C#

```csharp
String resFilePath = @"…";
int pageIndex = 0;
PDFWatermarkRes resWatermark = new PDFWatermarkFileRes(resFilePath, pageIndex);
```

VB.NET

```vbnet
Dim resFilePath As String = "…"
Dim pageIndex As Integer = 0
Dim resWatermark As PDFWatermarkRes = New PDFWatermarkImageRes(resFilePath, pageIndex)
```

Other Properties

If you want to display the watermark with different effects, you can set the effects with following properties:

Properties	Type	Description
IsAbovePage	Bool	Display the watermark above the page content or not. Default: false
Opacity	Float	Valid Value: 0.0F – 1.0F Default: 1.0F
Scale	Float	Scaling of resources adapting to page Valid Value: >= 0 0: Original size. Default: 0
Rotate	Float	The rotation angle of the resource (Counterclockwise) Unit: Degree Default: 0 degree
HoriAlignment		Horizontal alignment Default: WatermarkHoriAlignment.Center
HoriMargin		Horizontal offset Unit: inch Default: 0
VertAlignment		Vertical alignment Default: WatermarkVertAlignemnt.Center
VertMargin		Vertical offset Unit: inch Default: 0

Add Watermark to document

The following demo code will show you how to add watermark to page with specific settings.

```csharp
PDFDocument doc = new PDFDocument(inputFilePath);

PDFWatermarkTextRes resWatermark = new PDFWatermarkTextRes("Confidential", new Font("Arial", 72F, FontStyle.Regular), Color.Black, WatermarkTextAlignment.Center);

{
    //   define a watermark setting
    PDFPageWatermark watermark1 = new PDFPageWatermark(resWatermark);
    watermark1.IsAbovePage = false;
    watermark1.Rotate = -45F;
    watermark1.Opacity = 0.2F;

    //   define page range: all odd pages
    PageRangeOptions pageRange1 = new PageRangeOptions();
    pageRange1.AllPages = true;
    pageRange1.Subset = PageRangeSubset.Odd;

    // apply watermark settings to all odd pages
    PDFPageFieldHandler.ApplyWatermark(doc,  watermark1,  pageRange1);
}
{
    //   define a watermark setting
    PDFPageWatermark watermark2 = new PDFPageWatermark(resWatermark);
    watermark2.IsAbovePage = false;
    watermark2.Rotate = 45F;
    watermark2.Opacity = 0.2F;

    //   define page range: all even pages
    PageRangeOptions pageRange2 = new PageRangeOptions();
    pageRange2.AllPages = true;
    pageRange2.Subset = PageRangeSubset.Even;

    // apply watermark settings to all even pages
    PDFPageFieldHandler.ApplyWatermark(doc,  watermark2,  pageRange2);
}

doc.Save(outputFilePath);
```

```vbnet
VB.NET

Dim doc As PDFDocument = New PDFDocument("")
Dim resWatermark As PDFWatermarkTextRes = New
PDFWatermarkTextRes("Confidential", New Font("Arial", 72.0F,
FontStyle.Regular), Color.Black, WatermarkTextAlignment.Center)

'define a watermark setting
Dim watermark1 As PDFPageWatermark = New PDFPageWatermark(resWatermark)
watermark1.IsAbovePage = False
watermark1.Rotate = -45.0F
watermark1.Opacity = 0.2F

'define page range: all odd pages
Dim pageRange1 As PageRangeOptions = New PageRangeOptions()
pageRange1.AllPages = True
pageRange1.Subset = PageRangeSubset.Odd

'apply watermark settings to all odd pages
PDFPageFieldHandler.ApplyWatermark(doc, watermark1, pageRange1)
'define a watermark setting
Dim watermark2 As PDFPageWatermark = New PDFPageWatermark(resWatermark)
watermark2.IsAbovePage = False
watermark2.Rotate = 45.0F
watermark2.Opacity = 0.2F

'define page range: all even pages
Dim pageRange2 As PageRangeOptions = New PageRangeOptions()
pageRange2.AllPages = True
pageRange2.Subset = PageRangeSubset.Even

'apply watermark settings to all even pages
PDFPageFieldHandler.ApplyWatermark(doc, watermark2, pageRange2)
doc.Save("")
```

Related API(s) (**PDFPageFieldHandler.cs**):

```csharp
public static int ApplyWatermark(PDFDocument doc, PDFPageWatermark
watermark, PageRangeOptions ops);
```

Description:

Add watermark to PDF file with specified settings.

Parameters:

Name	Description	Valid Value
doc	Input pdf file	Can't be null

watermark	Watermark object	Can't be null
ops	Specific settings	-

Return:

Error code, 0 if success.

Remove Watermark from document

The following demo code will explain how to delete watermark from pdf document.

```C#
PDFDocument doc = new PDFDocument(inputFilePath);
PDFPageFieldHandler.RemoveWatermarks(doc);
doc.Save(outputFilePath);
```

```VB.NET
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
PDFPageFieldHandler. RemoveWatermarks(doc)
doc.Save(outputFilePath)
```

Related API(s) (**PDFPageFieldHandler.cs**):

```
public static int RemoveWatermarks(PDFDocument doc);
```

Description:

Remove watermarks from pdf document.

Parameters:

Name	Description	Valid Value
doc	Input pdf file	Can't be null

Return:

Error code, 0 if success.

```
public static int RemoveWatermarks(PDFDocument doc, int[] pageIndexs);
```

Description:

Remove specified page watermarks from pdf document.

Parameters:

Name	Description	Valid Value
doc	Input pdf file	Can't be null
pageIndexs	An array of page index of pages that will be deleted header/footer	0 ~ page number - 1

Return:

Error code, 0 if success.

Working with Document

Get Page Count

The following demo codes will show you how to get the total page number of PDF file.

```
C#

'Open a document.
String inputFilePath = "C:\\input.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get the total page number of the pdf file.
int pageNumber = doc.GetPageCount();
```

```
VB.NET

'Open a document.
Dim inputFilePath As String = "C:\\input.pdf"
Dim doc As PDFDocument = new PDFDocument(inputFilePath)
// Get the first page from the document.
Dim pageNumber As Integer = doc.GetPageCount()
```

Related API(s)(**PDFDocument.cs**):

```
public override int GetPageCount()
```

Description:

Get the total page number of the PDF file.

Return:

Total page number, 0 if failed.

Get Document Properties

You can do as follow to get the document type of input file:

```
C#

'Open a document.
String inputFilePath = "C:\\input.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get the total page number of the pdf file.
DocumentType type = doc.GetDocumentType();
```

Related API(s) (**PDFDocument.cs**):

```
public override DocumentType GetDocumentType()
```

Description:

      Get the document type of the input file.

Return:

      PDF, Invalid or other format if failed.

Insert/Add empty page(s) into a PDF file

To insert empty page(s) into PDF file, you can work as follows:

1. Open one existing PDF file through PDFDocument object.
2. Call the AddEmptyPage, AddEmptyPages, InsertPage or InsertPages to complete the inserting page(s) to a PDF file.
3. Call the PDFDocument object's Save/SaveToStream/SaveToBytes method and save the PDF object to file/stream.

The following demo codes will show you how to do:

```
C#

'Open a document.
String inputFilePath = "C:\\input.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Insert an empty page at 2 (previous to the third page).
doc.AddEmptyPage(2);
// Insert two empty pages at 2 (previous to the third page).
doc.AddEmptyPage(2,2);
// Save the file.
doc.Save("C:\\output.pdf");
```

Related API(s) (**PDFDocument.cs**)

```
public int AddEmptyPage(int pageIndex)
```

Description:

Add a blank page to PDF file at specified position.

Parameters:

Name	Description	Valid Value
pageIndex	The position of the inserted blank page.	0 ~ page number - 1, which is starting at 0.

Return:

Error code, 0 if success.

```
public int AddEmptyPage(int pageIndex, PaperSize pageSize)
```

Description:

Add a blank page with specified size to PDF file at specified position.

Parameters:

Name	Description	Valid Value
pageIndex	The position of the inserted blank page.	0 ~ page number - 1, which is starting at 0.
pageSize	The size of the inserted blank page	The value listed in the PaperSize.cs

Return:

Error code, 0 if success.

```
public int AddEmptyPages(int startPageIndex, int count)
```

Description:

Add continuous blank pages to PDF file at specified position.

Parameters:

Name	Description	Valid Value
startPageIndex	The position of the inserted blank page.	0 ~ page number - 1, which is starting at 0.

count	The count of inserted blank pages	Minimum is 1.

Return:

    Error code, 0 if success.

public int AddEmptyPages(int startPageIndex, int count, PaperSize pageSize)

Description:

    Add continuous blank pages with specified size to PDF file at specified position.

Parameters:

Name	Description	Valid Value
startPageIndex	The position of the inserted blank page.	0 ~ page number - 1, which is starting at 0.
count	The count of inserted blank pages	Minimum is 1.
paperSize	The size of the inserted blank page	The value listed in the PaperSize.cs

Return:

    Error code, 0 if success.

public void InsertPage(int pageIdx)

Description:

    Add a blank page to PDF file at specified position.

Parameters:

Name	Description	Valid Value
pageIdx	The position of the inserted blank page.	0 ~ page number - 1, which is starting at 0.

Return:

    Error code, 0 if success.

public void InsertPage(int pageIdx, PaperSize paperSize)

Description:

    Add a blank page with specified size to PDF file at specified position.

Parameters:

Name	Description	Valid Value
pageIdx	The position of the inserted blank page.	0 ~ page number - 1, which is starting at 0.
pageSize	The size of the inserted blank page	The value listed in the PaperSize.cs

Return:

    Error code, 0 if success.

Insert PDF Page(s) into another PDF file

To insert one PDF page into/from another PDF file, the following steps will work:

1. Open first PDF file with PDFDocument object.
2. Open second PDF file with PDFDocument object.
3. Get one page from first PDFDocument.
4. Call the method InsertPage to complete inserting page.

C#

```csharp
String inputFilePath1 = "C:\\1.pdf";
String inputFilePath2 = "C:\\2.pdf";
String outPutFilePath = "C:\\Output.pdf";
PDFDocument doc1 = new PDFDocument(inputFilePath1);
PDFDocument doc2 = new PDFDocument(inputFilePath2);
// Get a page from the first document.
PDFPage page = (PDFPage)doc1.GetPage(0);
// Specify a position for inserting the selected page.
int pageIndex = 2;
// Insert the page to the second document at specified position.
doc2.InsertPage(page, pageIndex);
// Output the new document.
doc2.Save(outPutFilePath);
```

VB.NET

```vbnet
Dim inputFilePath1 As String = "C:\\1.pdf"
Dim inputFilePath2 As String = "C:\\2.pdf"
Dim outPutFilePath As String = "C:\\Output.pdf"
Dim doc1 As PDFDocument = New PDFDocument(inputFilePath1)
Dim doc2 As PDFDocument = New PDFDocument(inputFilePath2)
'Get a page from the first document.
Dim page As PDFPage = doc1.GetPage(0)
'Specify a position (third page) for inserting the selected page.
Dim pageIndex As Integer = 2
'Insert the page to the second document at specified position.
doc2.InsertPage(page, pageIndex)
'Output the new document.
doc2.Save(outPutFilePath)
```

To insert PDF pages into/from another PDF file, the following steps will work:

1. Open first PDF file with PDFDocument object.
2. Open second PDF file with PDFDocument object.
3. Get pages from first PDFDocument.

4. Call the method InsertPage to complete inserting page.

C#

```csharp
String inputFilePath1 = "C:\\1.pdf";
String inputFilePath2 = "C:\\2.pdf";
String outPutFilePath = "C:\\Output.pdf";
PDFDocument doc1 = new PDFDocument(inputFilePath1);
PDFDocument doc2 = new PDFDocument(inputFilePath2);
// Set the page indexes that will be copied.
// The page indexes should be 0 ~ (page number of first document – 1)
int [] pageIndexes = new int[]{0, 2, 3};
// Get a page from the first document.
BasePage[] pages = doc1.DuplicatePage(pageIndexes);
// Specify a position for inserting the selected page.
int pageIndex = 2;
// Insert the page to the second document at specified position.
doc2.InsertPages(pages, pageIndex);
// Output the new document.
doc2.Save(outPutFilePath);
```

VB.NET

```vbnet
Dim inputFilePath1 As String = "C:\\1.pdf"
Dim inputFilePath2 As String = "C:\\2.pdf"
Dim outPutFilePath As String = "C:\\Output.pdf"
Dim doc1 As PDFDocument = New PDFDocument(inputFilePath1)
Dim doc2 As PDFDocument = New PDFDocument(inputFilePath2)
'Set the page indexes(first, third, fourth) that will be copied.
'The page indexes should be 0 ~ (page number of first document – 1)
Dim pageIndexes As Integer()= New Integer(){0, 2, 3};
'Get a page from the first document.
Dim pages As BasePage()= doc1.DuplicatePage(pageIndexes);
'Specify a position for inserting the selected page.
Dim pageIndex As int = 2;
'Insert the page to the second document at specified position.
doc2.InsertPages(pages, pageIndex);
// Output the new document.
doc2.Save(outPutFilePath);
```

If you want to get more demo codes, please refer to following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-insert/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-insert/

Related API(s) (**PDFDocument.cs**):

`public override void AddPage(`BasePage` page)`

Description:

      Add a PDF page into PDF file at default position, first page number.

Parameters:

Name	Description	Valid Value
page	A PDFPage object	Can't be null

`public override void AddPages(`BasePage`[] pages)`

Description:

      Add PDF pages into PDF file at default position, first page number.

Parameters:

Name	Description	Valid Value
pages	An array of PDFPage object	Can't be null

`public override void InsertPage(`BasePage` page, `int` pageIdx)`

Description:

      Add a PDF page into PDF file at specified position.

Parameters:

Name	Description	Valid Value
page	A PDFPage object	Can't be null
pageIdx	The specified position of inserted page	0 ~ page number - 1, which is starting at 0.

`public override void InsertPages(`BasePage`[] pages, `int` pageIdx)`

Description:

      Add PDF pages into PDF file at specified position.

Parameters:

Name	Description	Valid Value
pages	An array of PDFPage object	Can't be null
pageIdx	The specified position of inserted page	0 ~ page number - 1, which is starting at 0.

Delete PDF Page(s)

To delete/remove page(s) from PDF file, the following steps will work:

1. Open an existing PDF file.
2. Call the method DeletePage to delete specified page.
3. Call the method Save/SaveToBytes/SaveToStream to save the file to disk/stream.

The following demo code will show how to complete the deleting page(s):

```
C#

String filepath = @"C:\\input.pdf";
String outPutFilePath = @"C:\\output.pdf";
PDFDocument doc = new PDFDocument(filepath);
// Detele page 2 (actually the third page).
doc.DeletePage(2);
// Save the file.
doc.Save(outPutFilePath);
```

```
VB.NET

Dim filepath As String = "C:\\input.pdf"
Dim outPutFilePath As String = "C:\\output.pdf"
Dim doc As PDFDocument = New PDFDocument(filepath)
'Detele page 2 (actually the third page).
doc.DeletePage(2)
'Save the file.
doc.Save(outPutFilePath)
```

If you need more demo codes, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-delete/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-delete/

Related API(s) (**PDFDocument.cs**):

public override void DeletePage(int pageId)

Description:

Delete specified page from the input PDF file.

Parameters:

Name	Description	Valid Value
pageId	The page index of the deleted page	0 ~ page number - 1, which is starting at 0.

public override void DeletePages(int pageId, int pageCount)

Description:

Delete consecutive pages from the input PDF file starting at specified position.

Parameters:

Name	Description	Valid Value
pageId	The page index of the deleted page	0 ~ page number - 1, which is starting at 0.
pageCount	The total number of deleted pages	1 ~

```
public void DeletePages(int[] pageIndexes)
```

Description:

Delete specified pages from the input PDF file.

Parameters:

Name	Description	Valid Value
pageIndexes	An array of the page index of the deleted page	0 ~ page number - 1, which is starting at 0.

Get a Particular Page

To get a particular page from PDF file, the following demo code will be necessary:

```csharp
C#

String filepath = @"C:\\input.pdf";
PDFDocument doc = new PDFDocument(filepath);
// Get page 2 (actually the third page).
BasePage page = doc.GetPage(2);
```

```vbnet
VB.NET

Dim filepath As String = "C:\\input.pdf"
Dim doc As PDFDocument = New PDFDocument(filepath)
'Get page 2 (actually the third page).
Dim page As BasePage = doc.GetPage(2)
```

Related API(s) (**PDFDocument.cs**):

```
public override BasePage GetPage(int pageIdx)
```

Description:

Get specified page from the input PDF file.

Parameters:

Name	Description	Valid Value
pageIdx	The page index of page	0 ~ page number - 1

Return:

BasePage object, null if failed.

Replace PDF Page(s)

To replace PDF page(s), the following steps will be necessary:

1. Open first PDF file with PDFDocument object.
2. Open second PDF file.

3. Get one page from first PDF file.
4. Call the method UpdatePage to replace the second PDF file specified page.
5. Call the method Save/SaveToBytes/SaveToStream to save the file on the disk or stream.

The following demo codes will show you how to do:

C#

```csharp
//  load the PDF file that provides the page object
String resFilePath = "C:\\2.pdf";
PDFDocument resDoc = new PDFDocument(resFilePath);
//  get the 1st page in the document
PDFPage page = (PDFPage)resDoc.GetPage(0);
//  get PDFDocument object from a source file
String inputFilePath = "C:\\1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  replace the 3rd page by the PDFPage object
int pageIndex = 2;
doc.UpdatePage(page, pageIndex);
//  save the PDFDocument
String outputFilePath = "C:\\Output.pdf";
doc.Save(outputFilePath);
```

VB.NET

```vbnet
' load the PDF file that provides the page object
Dim resFilePath As String = "C:\\2.pdf"
Dim resDoc As PDFDocument = New PDFDocument(resFilePath)
' get the 1st page in the document
Dim page As PDFPage = resDoc.GetPage(0)
' get PDFDocument object from a source file
Dim inputFilePath As String = "C:\\1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
' replace the 3rd page by the PDFPage object
Dim pageIndex As Integer = 2
doc.UpdatePage(page, pageIndex)
' save the PDFDocument
Dim outputFilePath As String = "C:\\Output.pdf"
doc.Save(outputFilePath)
```

Also, you can refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-page-replace/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-page-replace/

Related API(s) (PDFDocument.cs):

```
public override int UpdatePage(BasePage newPage, int pageIdx)
```

Description:

>   Replace the specified page with new page.

Parameters:

Name	Description	Valid Value
newPage	The new page object	Can't be null
pageIdx	The position of the page that will be replaced	0 ~ page number - 1

Return:

>   Errod code, 0 if success.

```
public int UpdatePages(BasePage[] newPages, int[] pagesIdx)
```

Description:

>   Replace the specified pages with new pages.

Parameters:

Name	Description	Valid Value
newPages	The new page objects	Can't be null
pagesIdx	The page indexes of the page that will be replaced	0 ~ page number - 1

Return:

>   Errod code, 0 if success.

Swap Two Pages

The following demo code will show how to swap two pages of one PDF file:

C#

```csharp
String filepath = @" C:\\input.pdf ";
String outPutFilePath = @" C:\\outnput.pdf ";
PDFDocument doc = new PDFDocument(filepath);
// Swap first and second page.
doc.SwapTwoPages(0, 1);
//  Output the document
doc.Save(outputFilePath);
```

If you want to get more demo codes, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-sort/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-sort/

Related API(s) (**PDFDocument.cs**):

`public override void SwapTwoPages(int pageIndex1, int pageIndex2)`

Description:

Swap two specified pages.

Parameters:

Name	Description	Valid Value
pageIndex1	The first page index	0 ~ page number - 1
pageIndex2	The second page index	0 ~ page number - 1

`public void MovePage(int moveFrom, int moveTo)`

Description:

Swap two specified pages.

Parameters:

Name	Description	Valid Value
moveFrom	The first page index	0 ~ page number - 1
moveTo	The second page index	0 ~ page number - 1

Sort/Reorder PDF Pages

To reorder the PDF pages, you can do as follows:

1. Open an existing PDF file.
2. Set the reorder of the page indexes.
3. Call the method SortPage to complete the sorting pages.
4. Call the method Save to save PDFDocument object to file or stream.

The following demo code will help you achieve the sorting pages:

```
C#

// get PDFDocument object from a source file
String inputFilePath = "C:\\1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// show page count of the document
int pageCount = doc.GetPageCount();
// define the new order for all pages
// 1. the length of the array MUST BE equal to pageCount
// 2. each page index SHOULD be in the array and only once
// otherwise, the method would throw exception
int[] pageOrders = new int[] { 1, 3, 0, 5, 4, 6, 2 };
doc.SortPage(pageOrders);
// save the PDFDocument
String outputFilePath = "C:\\Output.pdf";
doc.Save(outputFilePath);
```

```
VB.NET

'get PDFDocument object from a source file
Dim inputFilePath As String = Program.RootPath + "\\" + "1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'show page count of the document
Dim pageCount As Integer = doc.GetPageCount()
'define the new order for all pages
'1. the length of the array MUST BE equal to pageCount
'2. each page index SHOULD be in the array and only once
'otherwise, the method would throw exception
Dim pageOrders = New Integer() {1, 3, 0, 5, 4, 6, 2}
doc.SortPage(pageOrders)
'save the PDFDocument
Dim outputFilePath As String = Program.RootPath + "\\" + "Output.pdf"
doc.Save(outputFilePath)
```

Related API(s)(**PDFDocument.cs**):

```
public override void SortPage(int[] pageOrders)
```

Description:

    Reorder the PDF pages with specified order.

Parameters:

Name	Description	Valid Value
pageOrders	New order of PDF pages	0 ~ page number - 1

Extract PDF Page(s) to PDF file/stream

To extract PDF page(s) to a file or stream, the following steps will be helpful:

1. Open an existing PDF file with PDFDocument object.
2. Define the page indexes will be extracted.
3. Call the method ExtractPages to extract pages and output to file or stream.

The following demo code will explain how to do:

C#

```csharp
// Get PDFDocument object from a source file.
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Select pages.
List<int> pageIndexes = new List<int>();
pageIndexes.Add(2);    // The 3rd page.
pageIndexes.Add(0);    // The 1st page.
pageIndexes.Add(3);    // The 4th page.
// Create the new document with 3 pages.
PDFDocument newDoc = (PDFDocument)doc.GetMultiDocument(pageIndexes);
// Save the PDFDocument.
String outputFilePath = Program.RootPath + "\\" + "Output.pdf";
newDoc.Save(outputFilePath);
```

VB.NET

```vbnet
'Get PDFDocument object from a source file.
Dim inputFilePath As String = Program.RootPath + "\\" + "1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Select pages.
Dim pageIndexes As New List(Of Integer)
pageIndexes.Add(2)  ' The 3rd page.
pageIndexes.Add(0)  ' The 1st page.
pageIndexes.Add(3)  ' The 4th page.
'Create the new document with 3 pages.
Dim newDoc As PDFDocument = doc.GetMultiDocument(pageIndexes)
'Save the PDFDocument.
Dim outputFilePath As String = Program.RootPath + "\\" + "Output.pdf"
newDoc.Save(outputFilePath)
```

Related API(s) (**PDFDocument.cs**):

```csharp
public override void ExtractPages(int[] pageIndexes, String outputFilePath)
```
Description:

Extract specified pages and save them into a new file.

Parameters:

Name	Description	Valid Value
pageIndexes	The page indexes that will be extracted	0 ~ page number - 1
outputFilePath	Output file path	Valid file path

```
public override void ExtractPages(int[] pageIndexes, Stream outputStream)
```
Description:

Extract specified pages and save them into a stream.

Parameters:

Name	Description	Valid Value
pageIndexes	The page indexed that will be extracted	0 ~ page number - 1
outputStream	Output stream	Valid FileStream or MemoryStream

Duplicate Page(s) from PDF file

To duplicate PDF page(s), you just need three steps:

1. Open an existing PDF file with PDFDocument object.
2. Define the page indexes will be copied.
3. Call the method DuplicatePage or DuplicatePages to copy the specified page(s).

The following demo code will work:

```csharp
C#
String filepath = @"C:\\input.pdf";
PDFDocument doc = new PDFDocument(filepath);
// Set the page indexes to copy.
// First, second, third page
int [] pageIndexes = new int[]{0, 1, 2};
// Copy the first page of PDF document.
PDFPage page = doc.DuplicatePage(1);
// Copy the specified pages of PDF document.
BasePage [] pages = doc.DuplicatePages(pageIndexes);
```

Related API(s) (**PDFDocument.cs**):

```
public override BasePage DuplicatePage(int pageIdx)
```

Description:

Copy specified page from the input PDF file.

Parameters:

Name	Description	Valid Value
pageIdx	The page index of page	0 ~ page number - 1

Return:

BasePage object, null if failed.

```
public BasePage[] DuplicatePages(int[] pageIndexes)
```

Description:

Copy specified pages from the input PDF file.

Parameters:

Name	Description	Valid Value
pageIndexes	The page indexes of pages that will be copied	0 ~ page number - 1

Return:

An array of BasePage object, null if failed.

Rotate PDF Page(s)

To rotate PDF page(s), you just need call the static method RotatePage or RotateAllPages directly.

```
C#

String inputFilePath = "C:\\1.pdf";
String outputFilePath = "C:\\Output.pdf";
// Specify the first page to be rotated.
int pageIndex = 0;
// Rotate 180 in clockwise.
int rotateInDegree = 180;
// Rotate the selected page.
PDFDocument.RotatePage(inputFilePath, pageIndex, rotateInDegree,
outputFilePath);
```

```
VB.NET

Dim inputFilePath As String = "C:\\1.pdf"
Dim outputFilePath As String = "C:\\Output.pdf"
' Specify the first page to be rotated.
Dim pageIndex As Integer = 0
' Rotate 180 in clockwise.
Dim rotateInDegree As Integer = 180
' Rotate the selected page.
PDFDocument.RotatePage(inputFilePath, pageIndex, rotateInDegree,
outputFilePath)
```

If you want to get more demo codes, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-rotate/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-rotate/

Related API(s) (**PDFDocument.cs**):

```
public static int RotatePage(String inputFilePath, int pageIndex, int
rotateInDegree, String outputFilePath)
```

Description:

Rotate the page with specified degree and save it to a new PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Input file path	Valid file path
pageIndex	The page index of the page that will be rotated	0 – page number – 1
rotateInDegree	The degree to rotate	90, 180, 270
outputFilePath	The output file path	Valid file path

Return:

Error code, 0 if success.

```
public static int RotatePage(String inputFilePath, int pageIndex,    int
```

```
rotateInDegree)
```

Description:

      Rotate the page with specified degree and cover the original PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Input file path	Valid file path
pageIndex	The page index of the page that will be rotated	0 – page number – 1
rotateInDegree	The degree to rotate	90, 180, 270

Return:

      Error code, 0 if success.

```
public static int RotateAllPages(String inputFilePath, int rotateInDegree,
String outputFilePath)
```

Description:

      Rotate all the pages with specified degree and save it to a new PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Input file path	Valid file path
rotateInDegree	The degree to rotate	90, 180, 270
outputFilePath	The output file path	Valid file path

Return:

      Error code, 0 if success.

```
public static int RotateAllPages(String inputFilePath, int rotateInDegree)
```

Description:

      Rotate the page with specified degree and cover the original PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Input file path	Valid file path
rotateInDegree	The degree to rotate	90, 180, 270

Return:

      Error code, 0 if success.

Split PDF File

With RasterEdge.XDoc.PDF library, you can achieve split PDF file by page number, file size, bookmark and page index.

Split by page number

Split PDF by page number, it will create files whose total page number is equal to each other and the page count is page number.

The following demo code will work:

C#

```csharp
String inputFilePath = "C:\\1.pdf";
// set split option
SplitOptions options = new SplitOptions(SplitMode.ByPage);
// limit the pages of each file to 8 pages
options.MaxPages = 8;
// set output option
SplitOutputOptions outputOps = new SplitOutputOptions();
outputOps.OutputFolder = "C:\\outputDirectory\\";
outputOps.Mode = 2;
outputOps.Label = @"Part";
outputOps.Separator = '_';
// split a PDF file with options
PDFDocument.SplitDocument(inputFilePath, options, outputOps);
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\1.pdf"
'Set split option
Dim options As SplitOptions = New SplitOptions(SplitMode.ByPage)
'Limit the pages of each file to 8 pages
options.MaxPages = 8
'Set output option
Dim outputOps As SplitOutputOptions = New SplitOutputOptions()
outputOps.OutputFolder = "C:\\outputDirectory\\"
outputOps.Mode = 2
outputOps.Label = "Part"
outputOps.Separator = "_"
'Split a PDF file with options
PDFDocument.SplitDocument(inputFilePath, options, outputOps)
```

If you want to try more demo code, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-split/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-split/

Related API(s) (**PDFDocument.cs**):

```
public static void SplitDocument(String inputFilePath, SplitOptions splitOp,
SplitOutputOptions outputOps)
```

Description:

Split PDF file according to the specified settings by split options.

Parameters:

Name	Description	Valid Value
inputFilePath	Input PDF file path	Valid PDF file path
splitOp	The split settings	-
outputOps	The output PDF files options	-

Split by file size

The following demo code will work:

```csharp
String inputFilePath = "C:\\1.pdf";
//set split option.
SplitOptions options = new SplitOptions(SplitMode.BySize);
//limit the size of each file to 0.03M.
options.MaxSize = 0.03F;
//set output option.
SplitOutputOptions outputOps = new SplitOutputOptions();
outputOps.OutputFolder = "C:\\outputDirectory\\";
outputOps.Mode = 1;
outputOps.Label = @"splitByFileSizePart";
outputOps.Separator = '_';
//split a PDF file with options.
PDFDocument.SplitDocument(inputFilePath, options, outputOps);
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\1.pdf"
'Set split option
Dim options As SplitOptions = New SplitOptions(SplitMode.BySize)
'Limit the size of each file to 0.1M bytes
options.MaxSize = 0.1F
'Set output option
Dim outputOps As SplitOutputOptions = New SplitOutputOptions()
outputOps.OutputFolder = "C:\\outputDirectory\\"
outputOps.Mode = 2
outputOps.Label = "Part"
outputOps.Separator = "_"
'Split a PDF file with options
PDFDocument.SplitDocument(inputFilePath, options, outputOps)
```

If you want to try more demo code, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-split/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-split/

Related API(s) (**PDFDocument.cs**):

public static void SplitDocument(String inputFilePath, SplitOptions splitOp,
SplitOutputOptions outputOps)

Description:

Split PDF file according to the specified settings by split options.

Parameters:

Name	Description	Valid Value
inputFilePath	Input PDF file path	Valid PDF file path
splitOp	The split settings	-
outputOps	The output PDF files options	-

Split by bookmark

The following demo code will work:

```csharp
C#

String inputFilePath = "C:\\1.pdf";
// Set split option
SplitOptions options = new SplitOptions(SplitMode.ByBookMark);
// Set output option
SplitOutputOptions outputOps = new SplitOutputOptions();
outputOps.OutputFolder = "C:\\outputDirectory\\";
outputOps.Mode = 2;
outputOps.Label = @"Part";
outputOps.Separator = '_';
// Split a PDF file with options
PDFDocument.SplitDocument(inputFilePath, options, outputOps);
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\1.pdf"
'Set split option
Dim options As SplitOptions = New SplitOptions(SplitMode.ByBookMark)
'Set output option
Dim outputOps As SplitOutputOptions = New SplitOutputOptions()
outputOps.OutputFolder = "C:\\outputDirectory\\"
outputOps.Mode = 2
outputOps.Label = "Part"
outputOps.Separator = "_"
'Split a PDF file with options
PDFDocument.SplitDocument(inputFilePath, options, outputOps)
```

If you want to try more demo code, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-split/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-split/

Related API(s) (**PDFDocument.cs**):

public static void SplitDocument(String inputFilePath, SplitOptions splitOp, SplitOutputOptions outputOps)

Description:

Split PDF file according to the specified settings by split options.

Parameters:

Name	Description	Valid Value
inputFilePath	Input PDF file path	Valid PDF file path
splitOp	The split settings	-
outputOps	The output PDF files options	-

Split by page index

To split PDF file by page index, calling the static method SplitDocument directly will work.

```csharp
C#

String inputFilePath = "C:\\source.pdf";
String firstFile = "C:\\Part_1.pdf";
String secondFile = "C:\\Part_2.pdf";
String[] outputFiles = new String[] { firstFile, secondFile };
//Split input document to two files
//File 0: page 0 ~ 1
//File 1: page 2 ~
PDFDocument.SplitDocument(inputFilePath, 2, outputFiles);
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\source.pdf"
Dim firstFile As String = "C:\\Part_1.pdf"
Dim secondFile As String = "C:\\Part_2.pdf"
Dim outputFiles As String() = New String(){ firstFile, secondFile }
'Split input document to two files
'File 0: page 0 ~ 1
'File 1: page 2 ~
PDFDocument.SplitDocument(inputFilePath, 2, outputFiles)
```

If you need more demo codes, please refer to the sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-split/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-split/

Related API(s) (**PDFDocument.cs**):

```
public static void SplitDocument(String source, int splitIndex, Stream[] streams)
```

Description:

Split the source PDF file into PDF files and save them into streams.

Parameters:

Name	Description	Valid Value
source	The source PDF file that will be splited.	An existing PDF file path
splitIndex	The split separator	1 ~ page number - 2
streams	An array of Stream object, the output streams to save the split files	Valid FileStream or MemoryStream

`public static void SplitDocument(String source, int splitIndex, String[] fileName)`

Description:

Split the source PDF file into PDF files and save them into files on the disk.

Parameters:

Name	Description	Valid Value
source	The source PDF file that will be splited.	An existing PDF file path
splitIndex	The split separator	1 ~ page number - 2
fileName	An array of file paths to save the split files	Valid file path

`public static void SplitDocument(Stream source, int splitIndex, Stream[] streams)`

Description:

Split the source PDF file into PDF files and save them into streams.

Parameters:

Name	Description	Valid Value
source	The source PDF file stream from other database or format	An valid PDF file data
splitIndex	The split separator	1 ~ page number - 2
streams	An array of Stream object, the output streams to save the split files	Valid FileStream or MemoryStream

`public static void SplitDocument(Stream source, int splitIndex, String[] fileName)`

Description:

Split the source PDF file into PDF files and save them into files on the disk.

Parameters:

Name	Description	Valid Value
source	The source PDF file stream from other database or format	An valid PDF file data
splitIndex	The split separator	1 ~ page number - 2
fileName	An array of file paths, the output streams to save the split files	Valid file path

`public static void SplitDocument(String inputFilePath, int[] splitIndexes, String[] outputFilePaths)`

Description:

Split the source PDF file into PDF files and save them into files on the disk.

Parameters:

Name	Description	Valid Value

inputFilePath	The source PDF file that will be split.	An existing PDF file path
splitIndexes	The split separator indexes	1 ~ page number - 2
outputFilePaths	An array of file paths, the output streams to save the split files	Valid file path

public static void SplitDocument(String inputFilePath, int[] splitIndexes, Stream[] outputStreams)

Description:

    Split the source PDF file into PDF files and save them into streams.

Parameters:

Name	Description	Valid Value
inputFilePath	The source PDF file that will be splited.	An existing PDF file path
splitIndexes	The split separator indexes	1 ~ page number - 2
outputStreams	An array of Stream object, the output streams to save the split files	Valid FileStream or MemoryStream

public static void SplitDocument(Stream inputStream, int[] splitIndexes, String[] outputFilePaths)

Description:

    Split the source PDF file into PDF files and save them into files on the disk.

Parameters:

Name	Description	Valid Value
inputStream	The source PDF file stream from other database or format	An valid PDF file data
splitIndexes	The split separator indexes	1 ~ page number - 2
outputFilePaths	An array of file paths, the output streams to save the split files	Valid file path

public static void SplitDocument(Stream inputFilePath, int[] splitIndexes, Stream[] outputStreams)

Description:

    Split the source PDF file into PDF files and save them into files on the disk.

Parameters:

Name	Description	Valid Value
inputFilePath	The source PDF file stream from other database or format	An valid PDF file data

splitIndexes	The split separator indexes	1 ~ page number - 2
outputStreams	An array of Stream object, the output streams to save the split files	Valid FileStream or MemoryStream

Combine/Append PDF Files

To combine PDF files into one PDF file, you just need to call the static CombineDocument directly.

C#

```csharp
String inputFilePath1 = "C:\\1.pdf";
String inputFilePath2 = "C:\\2.pdf";
String inputFilePath3 = "C:\\3.pdf";
String outputFilePath = "C:\\Output.pdf";
String[] inputFilePaths = new String[3] { inputFilePath1, inputFilePath2, inputFilePath3 };
// Combine three PDF files and output.
PDFDocument.CombineDocument(inputFilePaths, outputFilePath);
```

VB.NET

```vbnet
Dim inputFilePath1 As String = "C:\\1.pdf"
Dim inputFilePath2 As String = "C:\\2.pdf"
Dim inputFilePath3 As String = "C:\\3.pdf"
Dim outputFilePath As String = "C:\\Output.pdf"
Dim inputFilePaths = New String() {inputFilePath1, inputFilePath2, inputFilePath3}
' Combine three PDF files and output.
PDFDocument.CombineDocument(inputFilePaths, outputFilePath)
```

If you need to test more demo codes, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-merge/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-merge/

Related API(s) (**PDFDocument.cs**):

```
public   static   void   CombineDocument(Stream[]   strStreams,   String
outputFilePath)
```

Description:

Combine the source PDF streams into one PDF file and save it to a new PDF file on the disk.

Parameters:

Name	Description	Valid Value

srtStreams	Source PDF streams to be used to combine into one PDF File	Valid PDF streams.
outputFilePath	Output file path	Valid file path

public static void CombineDocument(Stream[] strStreams, Stream desStream)

Description:

Combine the source PDF streams into one PDF file and save it to a new stream.

Parameters:

Name	Description	Valid Value
srtStreams	Source PDF streams to be used to combine into one PDF File	Valid PDF streams.
desStream	Output PDF stream	Valid stream

public static void CombineDocument(String[] inputFilePaths, Stream desStream)

Description:

Combine the source PDF files into one PDF file and save it to a new stream.

Parameters:

Name	Description	Valid Value
inputFilePaths	Source PDF file paths to be used to combine into one file	Valid PDF file paths.
desStream	Output PDF stream	Valid stream

public static void CombineDocument(String[] inputFilePaths, String outputFilePath)

Description:

Combine the source PDF files into one PDF file and save it to a new stream.

Parameters:

Name	Description	Valid Value
inputFilePaths	Source PDF file paths to be used to combine into one file	Valid PDF file paths.
outputFilePath	Output file path	Valid file path

public override int AppendDocument(BaseDocument appendDoc)

Description:

Append the second PDF file to first one.

Parameters:

Name	Description	Valid Value
appendDoc	The second PDFDocument object	Can't be null

Return:

Error code, 0 if success.

Working with Annotations

The following table will describe Annotation Icons and the responding object in the library "RasterEdge.XDoc.PDF.dll".

Icon	Annotation Type	SDK Object
	Sticky note	PDFAnnotStickyNote
	Highlight text	PDFAnnotHighlight
	Add text comment	PDFAnnotText
	Attach file	PDFAnnotFileAttach
	Record audio	PDFAnnotSound
	Add stamp	PDFAnnotStamp
	Insert text at cursor	PDFAnnotTextInsert
	Add note to replace text	PDFAnnotTextReplace
	Strike through text	PDFAnnotDeleteLine
	Underline text	PDFAnnotUnderLine
	Add note to text	PDFAnnotHightlight
	Enable text correction keyboard shortcuts	
	Add text box	PDFAnnotTextBox
	Draw text callout	PDFAnnotTextCallout
	Draw line	PDFAnnotLine
	Draw arrow	PDFAnnotArrow
	Draw oval	PDFAnnotEllispe
	Draw rectangle	PDFAnnotRectangle
	Draw cloud	Not Supported

	Draw polygon	PDFAnnotPolygon
	Draw connected lines	PDFAnnotPolyline
	Draw free form	PDFAnnotPen
	Erase drawing	Not Supported

Annotation Type

Sticky Note

	The default setting value
Icon	
Color	Color.Yellow
Opacity	100%
Subject	Remark
Comment	""

Highlighe Text

	The default setting value
Color	Color.Yellow
Opacity	100%
Subject	Highlight
Comment	""

Add Text Comment

	The default setting value
Color	Color.Black
Font	Helvetica
Font Size	12
Subject	Printer text
Comment	""

Remarks:

In Adobe Reader application, the properties(Artist, Subject, Locked) will be not able to modify, if the modify must through special windows.

Attach File

	The default setting value
Icon	
Color	Color.Blue
Opacity	100%
Subject	Attach file
Description	""
Name	File name
Comment	File name

Add note to Replace Text

	The default setting value
Color	Color.Blue
Opacity	100%
Subject	""
Comment	""

Remarks:

This annotation is implemented by using Delete Annotation and Text Annotation.

Strikethrough Text

	The default setting value
Color	Color.Red
Opacity	100%
Subject	Delete line
Comment	""

Underline Text

	The default setting value
Color	Color.LightGreen
Style	Straight
Opacity	100%
Subject	Underline
Comment	""

Remarks:

Style Options: Straight | Wave

Common Drawing

Add Text Box

	The default setting value
Style	Straight
Thickness	1 pixel
Fill Color	Color.White
Border Color	Color.Red
Opacity	100%
Subject	Text box
Comment	""

Remarks:

Comment Color: **ALWAYS** Color.Red

Draw Text Callout

	The default setting value
Line End	Hollow triangle
Style	Straight
Thickness	1 pixel
Fill Color	Color.White
Border Color	Color.Red
Opacity	100%
Subject	Note
Comment	""

Remarks:

Comment Color: **ALWAYS** Color.Black

Draw Line

	The default setting value
Start	-
End	-
Style	Straight
Thickness	1 pixel
Fill Color	No Color
Color	Color.Red
Opacity	100%
Subject	Line
Comment	""

Draw Arrow

	The default setting value
Start	Hollow triangle
End	-
Style	Straight
Thickness	1 pixel
Fill Color	No Color
Color	Color.Red
Opacity	100%
Subject	Line
Comment	""

Draw Oval

	The default setting value
Style	Straight
Thickness	1 pixel
Fill Color	No Color
Color	Color.Red
Opacity	100%
Subject	Ellipse
Comment	""

Draw Rectangle

	The default setting value
Style	Straight
Thickness	1 pixel
Fill Color	No Color
Color	Color.Red
Opacity	100%
Subject	Rectangle
Comment	""

Draw Polygon

	The default setting value

Style	Straight
Thickness	1 pixel
Fill Color	No Color
Color	Color.Red
Opacity	100%
Subject	Polygon
Comment	""

Draw Connected Lines

	The default setting value
Start	-
End	-
Style	Straight
Thickness	1 pixel
Fill Color	No Color
Color	Color.Red
Opacity	100%
Subject	Polygon lines
Comment	""

Draw Free Form

	The default setting value
Thickness	1 pixel
Color	Color.Red
Opacity	100%
Subject	Pen
Comment	""

Hyperlink Annotation

	The default setting value
Thickness	1 pixel
Color	Color.Red
Opacity	100%
Subject	Link
Comment	""

Add annotations

The following sample codes will show you how to add annotations to PDF page:

Add Stick note annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create a stick note annotation
PDFAnnotStickyNote annot = new PDFAnnotStickyNote();
annot.Content = "This is a stick note";
annot.Position = new PointF(100F, 100F);
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a stick note annotation
Dim annot As PDFAnnotStickyNote = New PDFAnnotStickyNote()
annot.Content = "This is a stick note"
annot.Position = New PointF(100.0F, 100.0F)
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Highlight text annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create a highlight text annotation
PDFAnnotHighlight annot = new PDFAnnotHighlight();
annot.StartPoint = new PointF(100F, 200F);
annot.EndPoint = new PointF(300F, 400F);
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a highlight text annotation
Dim annot As PDFAnnotHighlight = New PDFAnnotHighlight()
annot.StartPoint = New PointF(100.0F, 200.0F)
annot.EndPoint = New PointF(300.0F, 400.0F)
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Text annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create a text annotation
PDFAnnotText annot = new PDFAnnotText();
annot.Boundary = new RectangleF(400F, 500F, 300F, 80F);
annot.Content = @"This is a text annotation";
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a text annotation
Dim annot As PDFAnnotText = New PDFAnnotText()
annot.Boundary = New RectangleF(400.0F, 500.0F, 300.0F, 80.0F)
annot.Content = "This is a text annotation"
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Insert Text annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
//  create an insert text annotation
PDFAnnotTextInsert annot = new PDFAnnotTextInsert();
annot.Position = new PointF(200F, 500F);
annot.Content = @"This is a text annotation";
//  add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
//  save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create an insert text annotation
Dim annot As PDFAnnotTextInsert = New PDFAnnotTextInsert()
annot.Position = New PointF(200.0F, 500.0F)
annot.Content = "This is a text annotation"
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Replace Text annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create a replace text annotation
PDFAnnotTextReplace annot = new PDFAnnotTextReplace();
annot.StartPoint = new PointF(100F, 200F);
annot.EndPoint = new PointF(300F, 400F);
annot.Content = "This is the replace text.";
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a replace text annotation
Dim annot As PDFAnnotTextReplace = New PDFAnnotTextReplace()
annot.StartPoint = New PointF(100.0F, 200.0F)
annot.EndPoint = New PointF(300.0F, 400.0F)
annot.Content = "This is the replace text."
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Strikethrough Line annotation to PDF page:

```csharp
C#

String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
//  create a strike through line annotation
PDFAnnotDeleteLine annot = new PDFAnnotDeleteLine();
annot.StartPoint = new PointF(100F, 200F);
annot.EndPoint = new PointF(300F, 400F);
//  add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
//  save to a new file
doc.Save("C:\\output.pdf");
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a strike through annotation
Dim annot As PDFAnnotDeleteLine = New PDFAnnotDeleteLine()
annot.StartPoint = New PointF(100.0F, 200.0F)
annot.EndPoint = New PointF(300.0F, 400.0F)
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Underline annotation to PDF page:

```csharp
C#

String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  get the 2nd page
PDFPage page = (PDFPage)doc.GetPage(1);
//  create a underline annotation
PDFAnnotUnderLine annot = new PDFAnnotUnderLine();
annot.StartPoint = new PointF(200F, 400F);
annot.EndPoint = new PointF(400F, 600F);
//  add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
//  save to a new file
doc.Save("C:\\output.pdf");
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create underline annotation
Dim annot As PDFAnnotUnderLine = New PDFAnnotUnderLine()
annot.StartPoint = New PointF(200.0F, 400.0F)
annot.EndPoint = New PointF(400.0F, 600.0F)
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Textbox annotation to PDF page:

```csharp
C#

String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create a text box annotation
PDFAnnotTextBox annot = new PDFAnnotTextBox();
annot.Boundary = new RectangleF(100F, 100F, 400F, 300F);
annot.Opacity = 0.5;
annot.Content = "This is a text box annotation.";
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a textbox annotation
Dim annot As PDFAnnotTextBox = New PDFAnnotTextBox()
annot.Boundary = New RectangleF(100.0F, 100.0F, 400.0F, 300.0F)
annot.Opacity = 0.5
annot.Content = "This is a text box annotation."
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page,  annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Callout annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create a callout annotation
PDFAnnotTextCallout annot = new PDFAnnotTextCallout();
annot.StartPoint = new PointF(100F, 100F);
annot.Boundary = new RectangleF(400F, 500F, 300F, 80F);
annot.Content = "This is a text callout annotation";
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a callout annotation
Dim annot As PDFAnnotTextCallout = New PDFAnnotTextCallout()
annot.StartPoint = New PointF(100.0F, 100.0F)
annot.Boundary = New RectangleF(400.0F, 500.0F, 300.0F, 80.0F)
annot.Content = "This is a text callout annotation"
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Line annotation to PDF page:

```csharp
C#

String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create a line annotation
PDFAnnotLine annot = new PDFAnnotLine();
annot.Start = new PointF(100F, 100F);
annot.End = new PointF(200F, 200F);
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a line annotation
Dim annot As PDFAnnotLine = New PDFAnnotLine()
annot.Start = New PointF(100.0F, 100.0F)
annot.End = New PointF(200.0F, 200.0F)
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Arrow annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create an arrow annotation
PDFAnnotArrow annot = new PDFAnnotArrow();
annot.Start = new PointF(300F, 100F);
annot.End = new PointF(400F, 150F);
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create an arrow annotation
Dim annot As PDFAnnotArrow = New PDFAnnotArrow()
annot.Start = New PointF(300.0F, 100.0F)
annot.End = New PointF(400.0F, 150.0F)
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Oval (Ellipse) annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create ellipse annotation
PDFAnnotEllipse annot = new PDFAnnotEllipse();
annot.Location = new PointF(300, 300);
annot.Width = 100F;
annot.Height = 50F;
annot.Opacity = 0.5;
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create an ellipse annotation
Dim annot As PDFAnnotEllipse = New PDFAnnotEllipse()
annot.Location = New PointF(300, 300)
annot.Width = 100.0F
annot.Height = 50.0F
annot.Opacity = 0.5
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Rectangle annotation to PDF page:

```csharp
C#

String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
//  create rectangle annotation
PDFAnnotRectangle annot = new PDFAnnotRectangle();
annot.Location = new PointF(400, 400);
annot.Width = 50F;
annot.Height = 50F;
annot.Opacity = 0.5;
//  add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
//  save to a new file
doc.Save("C:\\output.pdf");
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create  a rectangle annotation
Dim annot As PDFAnnotRectangle = New PDFAnnotRectangle()
annot.Location = New PointF(400, 400)
annot.Width = 100.0F
annot.Height = 50.0F
annot.Opacity = 0.5
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Polygon annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
//  create rectangle annotation
PDFAnnotPolygon annot = new PDFAnnotPolygon();

annot.Points = new PointF[5] {  new PointF(50F, 450F),
                                new PointF(100F, 450F),
                                new PointF(120F, 480F),
                                new PointF(100F, 500F),
                                new PointF(50F, 500F)
};
annot.Opacity = 0.5;
//  add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
//  save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a polygon annotation
Dim annot As PDFAnnotPolygon = New PDFAnnotPolygon()
Dim points(5) As PointF
points(0) = New PointF(50.0F, 450.0F)
points(0) = New PointF(100.0F, 450.0F)
points(0) = New PointF(120.0F, 480.0F)
points(0) = New PointF(100.0F, 500.0F)
points(0) = New PointF(50.0F, 500.0F)
annot.Points = points
annot.Opacity = 0.5
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Connected Lines annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
//  create rectangle annotation
PDFAnnotPolyline annot = new PDFAnnotPolyline();

annot.Points = new PointF[] {   new PointF(250F, 450F),
                                new PointF(300F, 450F),
                                new PointF(320F, 480F),
                                new PointF(300F, 500F),
                                new PointF(250F, 500F)
};
annot.Opacity = 0.5;
//  add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
//  save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a polygon annotation
Dim annot As PDFAnnotPolyline = New PDFAnnotPolyline()
Dim points(5) As PointF
points(0) = New PointF(50.0F, 450.0F)
points(0) = New PointF(100.0F, 450.0F)
points(0) = New PointF(120.0F, 480.0F)
points(0) = New PointF(100.0F, 500.0F)
points(0) = New PointF(50.0F, 500.0F)
annot.Points = points
annot.Opacity = 0.5
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Add Hyperlink annotation to PDF page:

C#

```csharp
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);
// create a link annotation
PDFAnnotLink annot = new PDFAnnotLink();
annot.Uri = "http://www.google.com";
annot.Opacity = 0.5;
annot.SetBoundary(new RectangleF(240f, 300f, 200, 20f));
// add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot);
// save to a new file
doc.Save("C:\\output.pdf");
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'get the 1st page
Dim page As PDFPage = doc.GetPage(0)
'create a link annotation
Dim annot As PDFAnnotLink = New PDFAnnotLink()
annot.Uri = "http://www.google.com"
annot.Opacity = 0.5
annot.SetBoundary(new RectangleF(240f, 300f, 200, 20f))
'add annotation to the page
PDFAnnotHandler.AddAnnotation(page, annot)
'save to a new file
doc.Save("C:\\output.pdf")
```

Also, you can refer to the following demo code sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-annotate-sticky-note/
http://www.rasteredge.com/how-to/csharp-imaging/pdf-annotate-text-highlight/
http://www.rasteredge.com/how-to/csharp-imaging/pdf-annotate-text/
http://www.rasteredge.com/how-to/csharp-imaging/pdf-annotate-text-box/
http://www.rasteredge.com/how-to/csharp-imaging/pdf-annotate-drawing-markups/
http://www.rasteredge.com/how-to/csharp-imaging/pdf-annotating/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-annotate-sticky-note/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-annotate-text-highlight/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-annotate-text/

Related API(s) (**PDFAnnotHandler.cs**)

`public static int AddAnnotation(PDFPage page, IPDFAnnot annot)`

Description:

Add annotation to specified page.

Parameters:

Name	Description	Valid Value
page	PDF page object	Can't be null.
annot	PDF annotation object	Can't be null

Return:

Error code, 0 if success.

`public static int AddAnnotation(String filePath, IPDFAnnot annot)`

Description:

Add annotation to specified PDF file and cover the original PDF file.

Parameters:

Name	Description	Valid Value
filePath	Source PDF file path	Valid path on the disk.
annot	PDF annotation object	Can't be null

Return:

Error code, 0 if success.

`public static int AddAnnotation(String filePath, List<IPDFAnnot> annots)`

Description:

Add annotations to specified PDF file and cover the original PDF file.

Parameters:

Name	Description	Valid Value
filePath	Source PDF file path	Valid path on the disk.
annots	A list of PDF annotation object	Can't be null

Return:

Error code, 0 if success.

`public static int AddAnnotation(String fileInPath, IPDFAnnot annot, String fileOutPath)`

Description:

Add annotations to specified PDF file and save it to a new PDF file.

Parameters:

Name	Description	Valid Value
fileInPath	Source PDF file path	Valid path on the disk.
annot	A list of PDF annotation object	Can't be null
fileOutPath	Output pdf file path	Valid path on the disk.

Error code, 0 if success.

```
public static int AddAnnotation(String fileInPath, List<IPDFAnnot> annots,
String fileOutPath)
```

Description:

Add annotation to specified PDF file and save it to a new PDF file.

Parameters:

Name	Description	Valid Value
fileInPath	Source PDF file path	Valid path on the disk.
annots	PDF annotation object	Can't be null
fileOutPath	Output pdf file path	Valid path on the disk.

Return:

Error code, 0 if success.

Get annotations

To get annotations from PDF file, the following steps will be helpful:

1. Open one PDF file with PDFDocument object.
2. Call the method GetAllAnnotations to extract annotations.

The following demo code will show you how to extract all the annotations from PDF document:

```
C#

// Open PDF file
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get all the annotations in the PDF file.
List<IPDFAnnot> allAnnotation = PDFAnnotHandler.GetAllAnnotations(doc);
```

```
VB.NET

'Open PDF file
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Extract all the annotations in the PDF file
Dim annotations As List(Of IPDFAnnot) =
PDFAnnotHandler.GetAllAnnotations(doc)
```

If you want to try more demo code, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-annotating/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-annotating/

Related API(s) (**PDFAnnotHandler.cs**):

`public static List<IPDFAnnot> GetAllAnnotations(PDFDocument doc)`

Description:

Get all the annotations in the PDF file.

Parameters:

Name	Description	Valid Value
doc	The source PDFDocument object	Can't be null

Return:

A list of IPDFAnnot object, the count of the list is 0 if failed or doesn't have annotation.

`public static List<IPDFAnnot> GetAllAnnotations(PDFPage page)`

Description:

Get all the annotations in the specified PDF page

Parameters:

Name	Description	Valid Value
page	The source PDF Page object	Can't be null

Return:

A list of IPDFAnnot object, the count of the list is 0 if failed or doesn't have annotation.

`public static List<IPDFAnnot> GetAnnotations(PDFPage page, PointF position)`

Description:

Get all the annotations in the PDF page at the specified position.

Parameters:

Name	Description	Valid Value
page	The source PDF Page object	Can't be null
position	Specified location	One point on the page

Return:

A list of IPDFAnnot object, the count of the list is 0 if failed or doesn't have annotation.

`public static List<IPDFAnnot> GetAnnotations(PDFDocument doc, int pageIndex, PointF position)`

Description:

Get all the annotations in the PDF page at the specified position by setting specified page index.

Parameters:

Name	Description	Valid Value
doc	The source PDFDocument object	Can't be null
pageIndex	Specified page index	0 - page number - 1
position	Specified location	One point on the page

Return:

A list of IPDFAnnot object, the count of the list is 0 if failed or doesn't have annotation.

`public static List<IPDFAnnot> GetAnnotations(PDFPage page, RectangleF boundary)`

Description:

　　Get all the annotations in the PDF page at the specified rectangle.

Parameters:

Name	Description	Valid Value
page	The source PDF Page object	Can't be null
boundary	Specified rectangle	One rectangle on the page

Return:

　　A list of IPDFAnnot object, the count of the list is 0 if failed or doesn't have annotation.

```
public static List<IPDFAnnot> GetAnnotations(PDFDocument doc, int pageIndex,
RectangleF boundary)
```

Description:

　　Get all the annotations in the PDF page at the specified rectangle by setting specified page index.

Parameters:

Name	Description	Valid Value
doc	The source PDFDocument object	Can't be null
pageIndex	Specified page index	0 - page number - 1
boundary	Specified rectangle	One rectangle on the page

Return:

　　A list of IPDFAnnot object, the count of the list is 0 if failed or doesn't have annotation.

Delete annotations

To delete annotation(s) in the PDF file, the following steps will work:

1. Open PDF document.
2. Call the method GetAllAnnotations to get all the annotations.
3. Call the method DeleteAnnotation to delete the specified annotation or annotations.
4. The PDFDocument object's method Save will allow you to save the file.

Please try the demo code to delete annotation(s) in the PDF file:

```csharp
C#

// Open PDF file
String inputFilePath = "C:\\demo_1.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Get all the annotations in the PDF file.
List<IPDFAnnot> allAnnotation = PDFAnnotHandler.GetAllAnnotations(doc);
// Delete all the annotations in the PDF file.
PDFAnnotHandler.DeleteAnnotation(doc, allAnnotation);
doc.Save("C:\\output.pdf");
```

Related API(s) (**PDFAnnotHandler.cs**):

`public static int DeleteAnnotation(PDFDocument doc, IPDFAnnot annot)`

Description:

Delete specified annotation in the PDF file.

Parameters:

Name	Description	Valid Value
doc	Target PDFDocument object	Can't be null
annot	The annotation object	Can't be null

Return:

Error code, 0 if success.

`public static int DeleteAnnotation(PDFDocument doc, List<IPDFAnnot> annots)`

Description:

Delete specified annotations in the PDF file.

Parameters:

Name	Description	Valid Value
doc	Target PDFDocument object	Can't be null
annots	A list of annotation object	Can't be null

Return:

Error code, 0 if success.

`public static int DeleteAnnotation(String inputFilePath, int pageIndex, PointF position, String outPutFilePath)`

Description:

Delete specified annotation(s) at specified page point in the PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Source PDF file	Valid file path on the disk.
pageIndex	Target page index to delete annotations	0 - page number - 1
position	The location of the deleted	A point at the page

	annotation	
outPutFilePath	The new PDF file output path	Valid file path on the disk.

Return:

Error code, 0 if success.

Modify annotations

Working with Forms

With RasterEdge.XDoc.PDF library, you can extract, fill, modify, add and delete form field freely.

The supported form filed type are as follows:

- Button
- CheckBox
- RadioButton
- TextBox
- ListBox
- ComboBox
- Signature

And the corresponding SDK object:

- AFButton
- AFCheckBox
- AFRadioButton
- AFTextBox
- AFListBox
- AFComboBox
- PDFSignature

Add/Insert form field

The following codes will show you how to add form field to PDF file:

C#

```csharp
//Open PDF document
PDFDocument doc = new PDFDocument("C:\\demo_1.pdf");
List<BaseFormField> fields = new List<BaseFormField>();
//  add a radio button field with default setting
AFRadioButton field1 = new AFRadioButton("AF_RadioButton_01");
field1.PageIndex = 0;
field1.Position = new PointF(100F, 100F);
fields.Add(field1);
//  add a checkbox field with default setting
AFCheckBox field2 = new AFCheckBox("AF_CheckBox_01");
field2.PageIndex = 0;
field2.Position = new PointF(300F, 100F);
fields.Add(field2);
//  add a checkbox field with default setting
AFTextBox field3 = new AFTextBox("AF_TextBox_01");
field3.PageIndex = 0;
field3.Position = new PointF(100F, 300F);
fields.Add(field3);
//  add a list box field with default setting
AFListBox field4 = new AFListBox("AF_ListBox_01");
field4.PageIndex = 0;
field4.Position = new PointF(100F, 500F);
field4.Items = new String[4] { "Item 1", "Item 2", "Item 3", "Item 4" };
fields.Add(field4);
//  add a combo box field with default setting
AFComboBox field5 = new AFComboBox("AF_ComboBox_01");
field5.PageIndex = 0;
field5.Position = new PointF(300F, 500F);
field5.Items = new String[4] { "Item 1", "Item 2", "Item 3", "Item 4" };
fields.Add(field5);
//  add a button field with default setting
AFButton field6 = new AFButton("AF_Button_01");
field6.PageIndex = 0;
field6.Position = new PointF(100F, 700F);
fields.Add(field6);
//  add fields to the input file
PDFFormHandler.AddFormFields(doc, fields);
doc.Save("C:\\output.pdf");
```

```vbnet
VB.NET

'Open PDF file
Dim doc As PDFDocument = New PDFDocument("C:\\demo_1.pdf")
Dim fields As List(Of BaseFormField) = New List(Of BaseFormField)
'add a radio button field with default setting
Dim field1 As AFRadioButton = New AFRadioButton("AF_RadioButton_01")
field1.PageIndex = 0
field1.Position = New PointF(100.0F, 100.0F)
fields.Add(field1)
'add a checkbox field with default setting
Dim field2 As AFCheckBox = New AFCheckBox("AF_CheckBox_01")
field2.PageIndex = 0
field2.Position = New PointF(300.0F, 100.0F)
fields.Add(field2)
'add a checkbox field with default setting
Dim field3 As AFTextBox = New AFTextBox("AF_TextBox_01")
field3.PageIndex = 0
field3.Position = New PointF(100.0F, 300.0F)
fields.Add(field3)
'add a list box field with default setting
Dim field4 As AFListBox = New AFListBox("AF_ListBox_01")
field4.PageIndex = 0
field4.Position = New PointF(100.0F, 500.0F)
field4.Items = New String() {"Item 1", "Item 2", "Item 3", "Item 4"}
fields.Add(field4)
'add a combo box field with default setting
Dim field5 As AFComboBox = New AFComboBox("AF_ComboBox_01")
field5.PageIndex = 0
field5.Position = New PointF(300.0F, 500.0F)
field5.Items = New String() {"Item 1", "Item 2", "Item 3", "Item 4"}
fields.Add(field5)
'add a button field with default setting
Dim field6 As AFButton = New AFButton("AF_Button_01")
field6.PageIndex = 0
field6.Position = New PointF(100.0F, 700.0F)
fields.Add(field6)
'add fields to the input file
PDFFormHandler.AddFormFields(doc, fields)
doc.Save("C:\\output.pdf")
```

Also, you can refer to the demo code sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-form-field-edit/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-form-field-edit/

Related API(s)(**PDFFormHandler.cs**):

```
public static int AddFormField(PDFDocument doc, BaseFormField field)
```

Description:

Add a form to PDF file on specified page.

Parameters:

Name	Description	Valid Value
doc	Target PDFDocument object	Can't be null
field	Form field object	Can't be null

Return:

Error code, 0 if success.

```
public static int AddFormFields(PDFDocument doc, List<BaseFormField> fields)
```

Description:

Add form fields to PDF file on specified page.

Parameters:

Name	Description	Valid Value
doc	Target PDFDocument object	Can't be null
fields	A list of form field object	Can't be null

Return:

Error code, 0 if success.

```
public static int AddFormFields(String inputFilePath, List<BaseFormField> fields, String outputFilePath)
```

Description:

Add form fields to PDF file on specified page and save to a new PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Target PDFDocument object	Valid file path on the disk
fields	A list of form field object	Can't be null
outputFilePath	Output new PDF file path	Valid file path on the disk

Return:

Error code, 0 if success.

Extract/Get form fields

The following codes will allow you get all the form fields in the PDF file:

```
C#
String inputFilePath = "C:\\input.pdf";
String outputFilePath = "C:\\output.pdf";
//Open PDF document
PDFDocument doc = new PDFDocument(inputFilePath);
//Extract form fields
List<BaseFormField> fields = PDFFormHandler.GetFormFields(doc);
```

```
VB.NET
Dim inputFilePath As String = "C:\\input.pdf"
Dim outputFilePath As String = "C:\\output.pdf"
'Open PDF document
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Extract form fields
Dim fields As List(Of BaseFormField) = PDFFormHandler.GetFormFields(doc)
```

Related API(s)(**PDFFormHandler.cs**):

public static List<BaseFormField> GetFormFields(PDFDocument doc)

Description:

Extract form fields from PDF file.

Parameters:

Name	Description	Valid Value
doc	Targer PDFDocument object	Can't be null

Return:

A list of form field object, null if failed.

public static List<BaseFormField> GetFormFields(String inputFilePath)

Description:

Extract form fields from specified PDF file path.

Parameters:

Name	Description	Valid Value
inputFilePath	Source PDF file path	Valid file path on the disk

Return:

A list of form field object, null if failed.

public static BaseFormField GetFormField(PDFDocument doc, int pageIndex, PointF position)

Description:

Extract form field from specified PDF file page at special position.

Parameters:

Name	Description	Valid Value

doc	Targer PDFDocument object	Can't be null
pageIndex	Target PDF page index	0 - page number - 1
position	A point on the page	-

Return:

A form field object, null if failed.

```csharp
public static BaseFormField GetFormField(String inputFilePath, int pageIndex, PointF position)
```

Description:

Extract form field from specified PDF file page at special position.

Parameters:

Name	Description	Valid Value
inputFilePath	Source PDF file path	Valid file path on the disk
pageIndex	Target PDF page index	0 - page number - 1
position	A point on the page	-

Return:

A form field object, null if failed.

Delete/Remove form fields

The following code will descript how to delete form field from PDF file:

```csharp
C#

String inputFilePath = "C:\\input.pdf";
String outputFilePath = "C:\\output.pdf";
// delete a field at position [110, 310] in page 1 (page index 0)
int pageIndex = 0;
PointF pos = new PointF(100F, 300F);
// remove the field and output the new document
int errCode = PDFFormHandler.RemoveFormField(inputFilePath, pageIndex, pos, outputFilePath);
if (errCode == 0)
{
    Console.WriteLine("Success");
}
else
{
    Console.WriteLine("Failed");
}
```

```
VB.NET
Dim inputFilePath As String = "C:\\input.pdf"
Dim outputFilePath As String = "C:\\output.pdf"
'remove a field by name and output the new document
Dim fieldName As String = "AF_RadioButton_01"
Dim errCode As Integer = PDFFormHandler.RemoveFormField(inputFilePath,
fieldName, outputFilePath)
If errCode = 0 Then
    Console.WriteLine("Success")
Else
    Console.WriteLine("Failed")
End If
```

Also, you can refer to the demo code sites:

Related API(s)(**PDFFormHandler.cs**):

```
public static int RemoveFormField(String inputFilePath, int pageIndex,
PointF position, String outputFilePath)
```

Description:

Delete form field at specified position in one PDF page and save to a new PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Target PDFDocument object	Valid file path on the disk
pageIndex	Target page index	0 - page number - 1
position	A point on the PDF page	-
outputFilePath	Output new PDF file path	Valid file path on the disk

Return:

Error code, 0 if success.

```
public static int RemoveFormField(String inputFilePath, BaseFormField field,
String outputFilePath)
```

Description:

Delete specified form field from PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Target PDFDocument object	Valid file path on the disk
field	Form field object	Can't be null
outputFilePath	Output new PDF file path	Valid file path on the disk

Return:

Error code, 0 if success.

```
public static int RemoveFormField(String inputFilePath, String name, String
outputFilePath)
```
Description:

    Delete specified form field from PDF file by form name and save to a new PDF file.

Parameters:

Name	Description	Valid Value
inputFilePath	Target PDFDocument object	Valid file path on the disk
name	Form field name	Can't be null
outputFilePath	Output new PDF file path	Valid file path on the disk

Return:

    Error code, 0 if success.

Modify/Update form fields

Related API(s)(**PDFFormHandler.cs**):

Fill field values

Related API(s)(**PDFFormHandler.cs**):
```
public static int FillFormField(String inputFilePath, int pageIndex, PointF
position, BaseUserInput inputInfo, String outputFilePath)
```

```
public static int FillFormField(PDFDocument doc, int pageIndex, PointF
position, BaseUserInput inputInfo)
```

```
public  static  int  FillFormField(String  inputFilePath,  String  name,
BaseUserInput inputInfo, String outputFilePath)
```

```
public static int FillFormField(PDFDocument doc, String name, BaseUserInput
inputInfo)
```

Working with Security and Signatures

Set file permissions

The following demo code will show you how to control the file permissions:

```csharp
String inputFilePath = "C:\\demo_1.pdf";
String outputFilePath = "C:\\output.pdf";
// Create a password setting object with user password "Hello World".
PasswordSetting passwordSetting = new PasswordSetting("Hello World");
// Set encryption level to AES-128.
passwordSetting.Level = EncryptionLevel.AES_128bit;
// Printing is allowed.
passwordSetting.IsPrint = true;
// Print with high-resolution.
passwordSetting.IsHighReso = true;
// Document is allowed to be changed.
passwordSetting.IsModify = true;
// Annotation is allowed.
passwordSetting.IsAnnot = true;
// Form filling is allowed.
passwordSetting.IsFillForm = true;
// Content extraction is allowed.
passwordSetting.IsExtract = true;
// Copying is allowed.
passwordSetting.IsCopy = true;
// PDF document assembling is allowed.
passwordSetting.IsAssemble = true;
// Add password to the file.
PDFDocument.AddPassword(inputFilePath, outputFilePath,
passwordSetting);
```

VB.NET

```vbnet
Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim outputFilePath As String = "C:\\output.pdf"
'Create a password setting object with user password "Hello World".
Dim passwordSetting As PasswordSetting = New PasswordSetting("Hello World")
'Set encryption level to AES-128.
passwordSetting.Level = EncryptionLevel.AES_128bit
'Printing is allowed.
passwordSetting.IsPrint = True
'Print with high-resolution.
passwordSetting.IsHighReso = True
'Document is allowed to be changed.
passwordSetting.IsModify = True
'Annotation is allowed.
passwordSetting.IsAnnot = True
'Form filling is allowed.
passwordSetting.IsFillForm = True
'Content extraction is allowed.
passwordSetting.IsExtract = True
'Copying is allowed.
passwordSetting.IsCopy = True
'PDF document assembling is allowed.
passwordSetting.IsAssemble = True
'Add password to the file.
PDFDocument.AddPassword(inputFilePath, outputFilePath, passwordSetting)
```

Modify file password

The following code will allow you to change password of PDF file:

C#

```csharp
// Define input and output file path.
String intputFilePath = "C:\\input.pdf";
String outputFilePath = "C:\\output.pdf";
// Set PDF passwords.
String userPassword = "you";
String newUserPassword = "fill";
String newOwnerPassword = "watch";
// Create setting for the new password.
PasswordSetting setting = new PasswordSetting(newUserPassword,
newOwnerPassword);
// Change password for an encrypted PDF file and output to a new file.
PDFDocument.ChangePassword(intputFilePath, outputFilePath,
userPassword, setting);
```

VB.NET

```vbnet
'Define input and output file path.
Dim intputFilePath As String = "C:\\input.pdf"
Dim outputFilePath As String = "C:\\output.pdf"
'Set PDF passwords.
Dim userPassword As String = "you"
Dim newUserPassword As String = "fill"
Dim newOwnerPassword As String = "watch"
'Create setting for the new password.
Dim setting As PasswordSetting = New PasswordSetting(newUserPassword,
newOwnerPassword)
'Change password for an encrypted PDF file and output to a new file.
PDFDocument.ChangePassword(intputFilePath, outputFilePath,
userPassword, setting)
```

Related API(s) (**PDFDocument.cs**)

```
public static int ChangePassword(String filePath, String password,
PasswordSetting newPasswordSetting)
```

Description:

   Modify the password of PDF file.

Parameters:

Name	Description	Valid Value
filePath	PDF file path	Valid file path on the disk

| password | Password to access PDF file | -- |
| newPasswordSetting | PasswordSetting object | Can't be null |

Return:

Error code, 0 if success.

public static int ChangePassword(String filePath, String outFilePath, String password, PasswordSetting newPasswordSetting)

Description:

Modify the password of PDF file and save to a new PDF file.

Parameters:

Name	Description	Valid Value
filePath	PDF file path	Valid file path on the disk
outFilePath	Output new PDF file path	Valid file path on the disk
password	Password to access PDF file	--
newPasswordSetting	PasswordSetting object	Can't be null

Return:

Error code, 0 if success.

public static int ChangePassword(Stream srcStream, Stream desStream, String password, PasswordSetting newPasswordSetting)

Description:

Modify the password of a PDF document stream and save to a new stream.

Parameters:

Name	Description	Valid Value
srcStream	The source stream	Valid FileStream or MemoryStream
desStream	Target output stream	Valid FileStream or MemoryStream
password	Password to access PDF file	--
passwordSetting	PasswordSetting object	Can't be null

Return:

Error code, 0 if success.

Encrypt PDF

The following demo codes will show how to encrypt PDF file:

```csharp
C#

// Define input and output files path.
String inputFilePath = "C:\\demo_1.pdf";
String outputFilePath = "C:\\1_with_pw.pdf";
// Set passwords for user and owner.
String userPassword = "you";
String ownerPassword = "me";
// Create password setting.
PasswordSetting setting = new PasswordSetting(userPassword,
ownerPassword);
// Add password to plain PDF file and output a new file.
PDFDocument.AddPassword(inputFilePath, outputFilePath, setting);
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\demo_1.pdf"
Dim outputFilePath As String = "C:\\output.pdf"
'Set passwords for user and owner.
Dim userPassword As String = "you"
Dim ownerPassword As String = "me"
'Create password setting.
Dim passwordSetting As PasswordSetting = New
PasswordSetting(userPassword, ownerPassword)
'Add password to plain PDF file and output a new file.
PDFDocument.AddPassword(inputFilePath, outputFilePath, passwordSetting)
```

If you want to try more demo codes, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-edit-password/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-edit-password/

Related API(s) (**PDFDocument.cs**)

```
public    static    int    AddPassword(String    filePath,    PasswordSetting
passwordSetting)
```

Description:

Add password to plain PDF file.

Parameters:

Name	Description	Valid Value
filePath	The source file path	Valid file path on the disk
passwordSetting	PasswordSetting object	Can't be null

Return:

Error code, 0 if success.

public static int AddPassword(String filePath, String outFilePath, PasswordSetting passwordSetting)

Description:

Add password to plain PDF file and save to a new file path.

Parameters:

Name	Description	Valid Value
filePath	The source file path	Valid file path on the disk
outFilePath	Output file path	Valid file path on the disk
passwordSetting	PasswordSetting object	Can't be null

Return:

Error code, 0 if success.

public static int AddPassword(Stream srcStream, Stream desStream, PasswordSetting passwordSetting)

Description:

Add password to plain PDF file and save to a new stream.

Parameters:

Name	Description	Valid Value
srcStream	The source stream	Valid FileStream or MemoryStream
desStream	Target output stream	Valid FileStream or MemoryStream
passwordSetting	PasswordSetting object	Can't be null

Return:

Error code, 0 if success.

Decrypt PDF

The following demo code will explain how to decrypt PDF file:

```csharp
C#

// Define input and output files path.
String inputFilePath = "C:\\1_with_pw.pdf";
// Set passwords for user and owner.
String userPassword = "you";
PDFDocument.RemovePassword(inputFilePath, userPassword);
```

```vbnet
VB.NET

Dim inputFilePath As String = "C:\\demo_1.pdf"
'Set passwords for user and owner.
Dim userPassword As String = "you"
'Remove password from PDF file and save to a new file.
PDFDocument.RemovePassword(inputFilePath, passwordSetting)
```

If you want to try more demo codes, please refer to the following sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-edit-password/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-edit-password/

Related API(s) (**PDFDocument.cs**)

public static int RemovePassword(String filePath, String password)

Description:

Remove password from an encrypted PDF file.

Parameters:

Name	Description	Valid Value
filePath	The source file path	Valid file path on the disk
password	The PDF file password	-

Return:

Error code, 0 if success.

public static int RemovePassword(String filePath, String outFilePath, String password)

Description:

Remove password from an encrypted PDF file and save to a new file path.

Parameters:

Name	Description	Valid Value
filePath	The source file path	Valid file path on the disk
outFilePath	Output file path	Valid file path on the disk
password	The PDF file password	-

Return:

Error code, 0 if success.

```
public static int RemovePassword(Stream srcStream, Stream desStream, String
password)
```
Description:
      Add password to plain PDF file and save to a new stream.

Parameters:

Name	Description	Valid Value
srcStream	The source stream	Valid FileStream or MemoryStream
desStream	Target output stream	Valid FileStream or MemoryStream
password	The PDF file password	-

Return:
      Error code, 0 if success.

Edit digital signatures

Please refer to the following sites:

      http://www.rasteredge.com/how-to/csharp-imaging/pdf-edit-digital-signature/
      http://www.rasteredge.com/how-to/vb-net-imaging/pdf-edit-digital-signature/

Redact text content

Please refer to the following sites:

      http://www.rasteredge.com/how-to/csharp-imaging/pdf-redact-text/
      http://www.rasteredge.com/how-to/vb-net-imaging/pdf-redact-text/

Redact images

Please refer to the following sites:

      http://www.rasteredge.com/how-to/csharp-imaging/pdf-redact-image/
      http://www.rasteredge.com/how-to/vb-net-imaging/pdf-redact-image/

Redact pages

Please refer to the following sites:

      http://www.rasteredge.com/how-to/csharp-imaging/pdf-redact-page/
      http://www.rasteredge.com/how-to/vb-net-imaging/pdf-redact-page/

Working with Stamp

With XDoc.PDF library, it is necessary to add PDFStampTemplate through PDFStampTemplateMgr before adding Stamp Annotation to pdf document. Every template has a unique template ID. When adding Stamp Annotation, it will create Stamp Annotation object with ID and responding User Info.

Create a Stamp Template

To create a stamp template, the following things are necessary:
- Background image, which will be provided as vector image
- Dynamic Fields, which is responding to User Info properties.

What's more, it has two ways to add background image for Stamp Template:
- Add by PDFContext
- Add from existing PDF file

The properties of Dynamic Field are as follows:
- The area of Stamp Template , RectangleF
- Field Type, which is responding to PDFAnnotStampUserInfo property.
- Font, Size, Color

The following codes will show you how to add by PDFContext (**Sample code 1**):

```csharp
C#                          createStampTemplateFromContext1(){}

PDFContext ctx = new PDFContext();
//  Set template size, it is just for designing stamp template, not the size
of stamp annotation.
ctx.SetWidth(new RELength(1, Units.IN));
ctx.SetHeight(new RELength(1, Units.IN));
ctx.DrawEllipse(new REPen(Color.Red, 5F), 3F, 3F, 90F, 90F);
ctx.DrawEllipse(new REPen(Color.Red, 1F), 7F, 7F, 82F, 82F);
ctx.DrawLine(new REPen(Color.Red, 1F), 7F, 48F, 89F, 48F);
PDFStampTemplate template = PDFStampTemplate.Create(ctx);
//  define a dynamic field for User Info - Name
PDFStampTemplateField field0 = new PDFStampTemplateField();
field0.Boundary = new RectangleF(0, 18, 100, 20);
field0.FieldType = PDFStampTemplateFieldType.Name;
field0.TextColor = Color.FromArgb(255, 0, 0);
field0.TextFont = new Font("Arial", 10, FontStyle.Regular);
template.AddField(field0);
//  define a dynamic field for User Info - Company
PDFStampTemplateField field1 = new PDFStampTemplateField();
field1.Boundary = new RectangleF(0, 62, 100, 20);
field1.FieldType = PDFStampTemplateFieldType.Company;
field1.TextColor = Color.FromArgb(255, 0, 0);
field1.TextFont = new Font("Arial", 8, FontStyle.Regular);
template.AddField(field1);
//  define a dynamic field for User Info - Current Date Time
PDFStampTemplateField field2 = new PDFStampTemplateField();
field2.Boundary = new RectangleF(0, 40, 100, 20);
field2.FieldType = PDFStampTemplateFieldType.CurrertDateTime;
field2.TextColor = Color.FromArgb(255, 0, 0);
field2.TextFont = new Font("Arial", 6, FontStyle.Regular);
template.AddField(field2);
```

```vbnet
VB.NET

Dim ctx As PDFContext = New PDFContext()
'Set template size, it is just for designing stamp template, not the size of
stamp annotation.
ctx.SetWidth(New RELength(1, Units.IN))
ctx.SetHeight(New RELength(1, Units.IN))
ctx.DrawEllipse(New REPen(Color.Red, 5.0F), 3.0F, 3.0F, 90.0F, 90.0F)
ctx.DrawEllipse(New REPen(Color.Red, 1.0F), 7.0F, 7.0F, 82.0F, 82.0F)
ctx.DrawLine(New REPen(Color.Red, 1.0F), 7.0F, 48.0F, 89.0F, 48.0F)


Dim template As PDFStampTemplate = PDFStampTemplate.Create(ctx)


'define a dynamic field for User Info - Name
Dim field0 As PDFStampTemplateField = New PDFStampTemplateField()
field0.Boundary = New RectangleF(0, 18, 100, 20)
field0.FieldType = PDFStampTemplateFieldType.Name
field0.TextColor = Color.FromArgb(255, 0, 0)
field0.TextFont = New Font("Arial", 10, FontStyle.Regular)
template.AddField(field0)


'define a dynamic field for User Info - Company
Dim field1 As PDFStampTemplateField = New PDFStampTemplateField()
field1.Boundary = New RectangleF(0, 62, 100, 20)
field1.FieldType = PDFStampTemplateFieldType.Company
field1.TextColor = Color.FromArgb(255, 0, 0)
field1.TextFont = New Font("Arial", 8, FontStyle.Regular)
template.AddField(field1)


'define a dynamic field for User Info - Current Date Time
Dim field2 As PDFStampTemplateField = New PDFStampTemplateField()
field2.Boundary = New RectangleF(0, 40, 100, 20)
field2.FieldType = PDFStampTemplateFieldType.CurrertDateTime
field2.TextColor = Color.FromArgb(255, 0, 0)
field2.TextFont = New Font("Arial", 6, FontStyle.Regular)
template.AddField(field2)
```

The following codes will show you how to add by PDFContext (**Sample code 2**):

```csharp
C#

createStampTemplateFromContext2(){}

PDFContext ctx = new PDFContext();
ctx.SetWidth(new RELength(2, Units.IN));
ctx.SetHeight(new RELength(1, Units.IN));
ctx.DrawRectangle(new REPen(Color.Red, 5F), 3F, 3F, 186F, 90F);
ctx.FillRectangle(new RESolidBrush(Color.LightGray), 7F, 7F, 178F, 82F);
ctx.DrawLine(new REPen(Color.Gray, 1F), 7F, 48F, 185F, 48F);
ctx.DrawRectangle(new REPen(Color.Red, 1F), 7F, 7F, 178F, 82F);


PDFStampTemplate template = PDFStampTemplate.Create(ctx);


PDFStampTemplateField field0 = new PDFStampTemplateField();
field0.Boundary = new RectangleF(0, 18, 200, 20);
field0.FieldType = PDFStampTemplateFieldType.Name;
field0.TextColor = Color.FromArgb(255, 0, 0);
field0.TextFont = new Font("Arial", 10, FontStyle.Regular);
//  set company field
PDFStampTemplateField field1 = new PDFStampTemplateField();
field1.Boundary = new RectangleF(0, 62, 200, 20);
field1.FieldType = PDFStampTemplateFieldType.Company;
field1.TextColor = Color.FromArgb(255, 0, 0);
field1.TextFont = new Font("Arial", 8, FontStyle.Regular);
//  set current time field
PDFStampTemplateField field2 = new PDFStampTemplateField();
field2.Boundary = new RectangleF(0, 40, 200, 20);
field2.FieldType = PDFStampTemplateFieldType.CurrertDateTime;
field2.TextColor = Color.FromArgb(255, 0, 0);
field2.TextFont = new Font("Arial", 6, FontStyle.Regular);
//
template.AddField(field0);
template.AddField(field1);
template.AddField(field2);
```

```vbnet
Dim ctx As PDFContext = New PDFContext()
ctx.SetWidth(New RELength(2, Units.IN))
ctx.SetHeight(New RELength(1, Units.IN))
ctx.DrawRectangle(New REPen(Color.Red, 5.0F), 3.0F, 3.0F, 186.0F, 90.0F)
ctx.FillRectangle(New RESolidBrush(Color.LightGray), 7.0F, 7.0F, 178.0F,
82.0F)
ctx.DrawLine(New REPen(Color.Gray, 1.0F), 7.0F, 48.0F, 185.0F, 48.0F)
ctx.DrawRectangle(New REPen(Color.Red, 1.0F), 7.0F, 7.0F, 178.0F, 82.0F)

Dim template As PDFStampTemplate = PDFStampTemplate.Create(ctx)

Dim field0 As PDFStampTemplateField = New PDFStampTemplateField()
field0.Boundary = New RectangleF(0, 18, 200, 20)
field0.FieldType = PDFStampTemplateFieldType.Name
field0.TextColor = Color.FromArgb(255, 0, 0)
field0.TextFont = New Font("Arial", 10, FontStyle.Regular)
'set company field
Dim field1 As PDFStampTemplateField = New PDFStampTemplateField()
field1.Boundary = New RectangleF(0, 62, 200, 20)
field1.FieldType = PDFStampTemplateFieldType.Company
field1.TextColor = Color.FromArgb(255, 0, 0)
field1.TextFont = New Font("Arial", 8, FontStyle.Regular)
'set current time field
Dim field2 As PDFStampTemplateField = New PDFStampTemplateField()
field2.Boundary = New RectangleF(0, 40, 200, 20)
field2.FieldType = PDFStampTemplateFieldType.CurrertDateTime
field2.TextColor = Color.FromArgb(255, 0, 0)
field2.TextFont = New Font("Arial", 6, FontStyle.Regular)

template.AddField(field0)
template.AddField(field1)
template.AddField(field2)
```

The following codes will explain how to create a stamp template from pdf file(**Sample Code 3**).

```csharp
String sourceFile = @"\StampBg1.pdf";        createStampTemplateFromFile(){}
int pageIndex = 0;
PDFStampTemplate template = PDFStampTemplate.Create(sourceFile, pageIndex);
```

Add Stamp Annotation

If you want to try adding Stamp Annotation, the following demo codes are enough:

```csharp
C#
internal static void testAnnotStamp()
{
    String outputFolder = "C:\\";
    // Init Stamp Template Manager
    initTemplate();
    List<String> ids = PDFStampTemplateMgr.GetTemplateIDs();
    String inputFilePath = outputFolder + "demo_1.pdf";
    String outputFilePath = outputFolder + "annotStamp.pdf";
    //  Create a stamp annotation
    PDFAnnotStamp annot = new PDFAnnotStamp();
    // select stamp template
    annot.StampTemplateID = aTemplateID;
    //  set annotation position and size
    annot.Boundary = new RectangleF(100, 100, 80, 40);
    //  set user info
    annot.UserInfo.Company = "Adobe Ltd.";
    annot.UserInfo.FamilyName = "Kitty";
    annot.UserInfo.GivenName = "Hello";
    // other annotation properties
    annot.Opacity = 1.0;
    annot.PageIndex = 0;
    //  add annotation
    PDFAnnotHandler.AddAnnotation(inputFilePath, annot, outputFilePath);
}
private static void initTemplate()
{
    // Clear all current Stamp Templates
    PDFStampTemplateMgr.Reset();
     //  Add Tempaltes
     PDFStampTemplateMgr.AddTemplate("Template1", createStampTemplateFromContext1(), true);
    PDFStampTemplateMgr.AddTemplate("Template2", createStampTemplateFromContext2(), true);
    //  To add more templates
}
private static PDFStampTemplate createStampTemplateFromContext1()
{
    //Copy Sample code 1 here.
}
private static PDFStampTemplate createStampTemplateFromContext2()
{
    //Copy Sample code 2 here.
}
private static PDFStampTemplate createStampTemplateFromFile()
{
    //Copy Sample code 3 here.
}
```

Working with Bookmarks

Bookmarks are held in the REOutline object's Entry collection, every bookmark is held in the OutLine object. You can get the bookmark object (OutLine) by calling the method GetOutline of PDFDocument object. If you want to have a try, please refer to "Extract/Get bookmarks".

Now, I will explain the information included in the OutLine object which is a bookmark.

A bookmark includes the following information:

    a.   Page index

    b.   Bookmark location

    c.   Bookmark level

    d.   Bookmark content/text

    e.   Parent bookmark

Extract/Get bookmarks

To retrieve bookmarks of PDF file, the following steps will be helpful:

1. Open on existing PDF file with PDFDocument object.
2. Call the method GetOutline to get the REOutline object including all the bookmarks, if the PDF file doesn't have any bookmark, the length of REOutline.Entry collection will be Zero.
3. In XDoc.PDF library, the OutLine object inherits the abstract class "Entry", you can get the information of one bookmark by calling the following methods:
   a) GetPageIndex, the page index of PDF page is starting at 0.
   b) GetLevel, the bookmark level is starting at 0.
   c) GetLocation, the return value is a PointF object including x,y coordinate.
   d) GetText, the bookmark content.

The following code will show you how to extract bookmarks of PDF file:

```
C#

// Retrieve PDF document outline.
// Open one PDF document
String inputFilePath = "C:\input.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Retrieve REOutline object including the bookmarks
REOutline outline = doc.GetOutline();
// Output all the bookmark information
foreach (REEntry entry in outline.Entry)
{
        Console.WriteLine("Page:     " + entry.GetPageIndex());
        Console.WriteLine("Level:    " + entry.GetLevel());
        Console.WriteLine("Position X,Y: " + entry.GetLocation().X +","+
entry.GetLocation().Y);
        Console.WriteLine("Text:     " + entry.GetText());
```

```vbnet
VB.NET

'Retrieve PDF document outline.
'Open one PDF document
Dim inputFilePath As String = "C:\\2.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Get the REOutline object including all the bookmarks entry.
Dim outline As REOutline = doc.GetOutline()
'Output all the bookmark information
For Each entry As REEntry In outline.Entry
    Console.WriteLine("Page:     {0}", entry.GetPageIndex())
    Console.WriteLine("Level:    {0}", entry.GetLevel())
    Console.WriteLine("Position: {0}", entry.GetLocation().X +","+
entry.GetLocation().Y)
    Console.WriteLine("Text:     {0}", entry.GetText())
Next
```

Related API(s) (**OutLine.cs**):

public override int GetPageIndex()

Description:

    Get the page index of the bookmark.

Returns:

    Page number, which is starting at 0.

public override int GetLevel()

Description:

    Get the bookmark level which is starting at 1.

Returns:

    Bookmark Level, starting at 1.

public override PointF GetLocation()

Description:

    Get the bookmark location.

Returns:

    PointF object including X,Y coordinate.

public override String GetText()

Description:

    Get the bookmark text.

Returns:

    Bookmark content which will display in the bookmark label.

public override REEntry GetParentREEntry()

Description:

Get the parent bookmark of current bookmark.

Returns:

REEntry object.

Modify bookmarks

To modify the bookmark, the following steps are enough:

1. Open en existing PDF file.
2. Get the REOutline object.
3. Reset the bookmark label information.
4. Update the text in the PDF page.
5. Call the method SetOutline to update the new bookmarks.
6. Call the method Save to output new PDF file.

The following codes will show how to modify the bookmark labels:

```csharp
C#

// Open a PDF document.
String inputFilePath = "C:\\input.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Retrieve PDF document outline.
REOutline outline = doc.GetOutline();
// Modify the bookmark labels.
for (int i = 0; i < outline.Entry.Count; i++ )
{
    OutLine singleOutline = (OutLine)outline.Entry[i];
    singleOutline.SetPageIndex(0);
    singleOutline.SetText("This bookmark was modified!!");
    singleOutline.SetLocation(new PointF(200F, 200F));
    singleOutline.SetLevel(2);
}
// Update the bookmark with the new values.
doc.SetOutline(outline);
// Save the modified PDF file to a new file.
doc.Save("C:\\modify_bookmark.pdf");
```

```vbnet
VB.NET
'Open one PDF file
Dim inputFilePath As String = "C:\\input.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Retrieve the PDF outline
Dim outline As REOutline = doc.GetOutline()
Dim i As Integer = 0
'Modify the bookmark labels
For i = 0 To outline.Entry.Count - 1
    Dim bookmark As OutLine = outline.Entry(i)
    bookmark.SetPageIndex(0)
    bookmark.SetText("This bookmark was modified!!")
    bookmark.SetLocation(New PointF(200.0F, 200.0F))
    bookmark.SetLevel(2)
Next i
'Update the bookmark labels
doc.SetOutline(outline)
'Save the modified file to a new one
doc.Save("C:\\modify_bookmark.pdf")
```

If you want to try more demo codes, please refer to the sites:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-edit-bookmark-outline/

http://www.rasteredge.com/how-to/vb-net-imaging/pdf-edit-bookmark-outline/

Related API(s) (**OutLine.cs**):

```csharp
public override void SetPageIndex(int pageIndex)
```

Description:

Set the bookmark new page index.

Parameters:

Name	Description	Valid Value
pageIndex	Bookmark new page index	0 ~ page number - 1

```csharp
public override void SetLocation(PointF location)
```

Description:

Set the bookmark new location on the page.

Parameters:

Name	Description	Valid Value
location	Bookmark new location	Any position on the page.

```csharp
public override void SetLevel(int level)
```

Description:

Set the bookmark new level.

Name	Description	Valid Value
level	Bookmark new level	1~

public override void SetText(String text)

Description:

Set the bookmark new content.

Parameters:

Name	Description	Valid Value
text	Bookmark new content	Any text string

public override void SetParentREEntry(REEntry entry)

Description:

Set the bookmark new content.

Parameters:

Name	Description	Valid Value
entry	Bookmark new parent bookmark	Can't be null

Add/Insert bookmarks

The following demo codes will allow you add a bookmark as the following image shows:
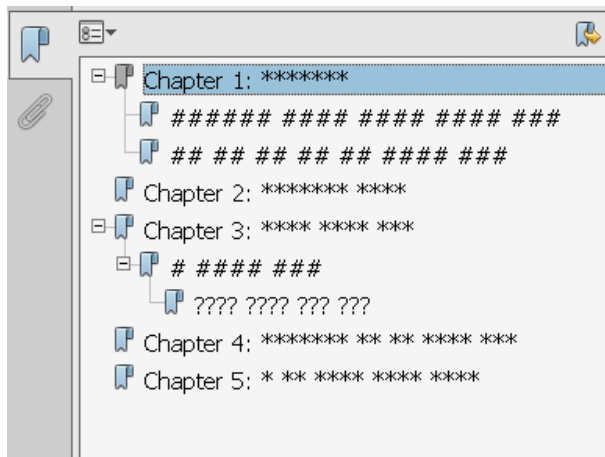
C#

```csharp
String inputFilePath = "C:\\input.pdf";
String outputFilePath = "C:\\output.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// Initial a new outline object.
REOutline outline = new REOutline();
```

```csharp
// Create 1st entry: Level = 1; Location: page index 0, y = 50.
OutLine entryChapter1 = new OutLine("Chapter 1: *******", 1, 0, 50);

// Add 1st entry to outline.
outline.Entry.Add(entryChapter1);

// Create 1st entry's child entries.
// Create 1st child entry: Level = 2; Location: page index 1, y = 0.
OutLine entryChapter11 = new OutLine("###### #### #### #### ###", 2, 1, 0);

// Connect this entry to the parent entry.
entryChapter11.SetParentREEntry(entryChapter1);

// Add 1st child of 1st entry to outline.
outline.Entry.Add(entryChapter11);

// Create 2nd child entry: Level = 2; Location: page index 2, y = 0.
OutLine entryChapter12 = new OutLine("## ## ## ## ## #### ###", 2, 2, 0);

// Connect this entry to the parent entry.
entryChapter12.SetParentREEntry(entryChapter1);

// Add 2nd child of 1st entry to outline.
outline.Entry.Add(entryChapter12);

// Create 2nd entry: Level = 1; Location: page index 3, y = 100.
OutLine entryChapter2 = new OutLine("Chapter 2: ******* ****", 1, 3, 100);

// Add 2nd entry to outline.
outline.Entry.Add(entryChapter2);

// Create 3rd entry: Level = 1; Location: page index 5, y = 200.
OutLine entryChapter3 = new OutLine("Chapter 3: **** **** ***", 1, 5, 200);

// Add 3rd entry to outline.
outline.Entry.Add(entryChapter3);

// Create 3rd entry's child entries.
// Create 1st child entry: Level = 2; Location: page index 9, y = 0.
OutLine entryChapter31 = new OutLine("# #### ###", 2, 9, 0);

// Connect this entry to the parent entry.
entryChapter31.SetParentREEntry(entryChapter3);
```

```
// Add 1st child of 3rd entry to outline.
outline.Entry.Add(entryChapter31);

// Create a child entry for the 1st child entry of the 3rd entry.
// Create entry: Level = 3; Location: page index 9, y = 500.
OutLine entryChapter311 = new OutLine("???? ???? ??? ???", 3, 9, 500);

// Connect this entry to the parent entry.
entryChapter311.SetParentREEntry(entryChapter31);

// Add 1st child of 3rd entry to outline.
outline.Entry.Add(entryChapter311);

// Create 4th entry: Level = 1; Location: page index 10, y = 0.
OutLine entryChapter4 = new OutLine("Chapter 4: ******* ** ** **** ***",
1, 10, 0);

// Add 4th entry to outline.
outline.Entry.Add(entryChapter4);

// Create 5th entry: Level = 1; Location: page index 20, y = 0.
OutLine entryChapter5 = new OutLine("Chapter 5: * ** **** **** ****", 1,
20, 0);

// Add 5th entry to outline.
outline.Entry.Add(entryChapter5);

// Update the new outline.
doc.SetOutline(outline);

doc.Save(outputFilePath);
```

VB.NET

```
Dim inputFilePath As String = "C:\\input.pdf"
Dim outputFilePath As String = "C:\\output.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)

' Initial a new outline object.
Dim outline As REOutline = New REOutline()

' Create 1st entry: Level = 1; Location: page index 0, y = 50.
Dim entryChapter1 As OutLine = New OutLine("Chapter 1: *******", 1, 0, 50)
```

```vbnet
' Add 1st entry to outline.
outline.Entry.Add(entryChapter1)

' Create 1st entry's child entries.
' Create 1st child entry: Level = 2; Location: page index 1, y = 0.
Dim entryChapter11 As OutLine = New OutLine("###### #### #### #### ###",
2, 1, 0)

' Connect this entry to the parent entry.
entryChapter11.SetParentREEntry(entryChapter1)

' Add 1st child of 1st entry to outline.
        outline.Entry.Add(entryChapter11)

' Create 2nd child entry: Level = 2; Location: page index 2, y = 0.
Dim entryChapter12 As OutLine = New OutLine("## ## ## ## ## #### ###", 2,
2, 0)

' Connect this entry to the parent entry.
entryChapter12.SetParentREEntry(entryChapter1)

' Add 2nd child of 1st entry to outline.
outline.Entry.Add(entryChapter12)

' Create 2nd entry: Level = 1; Location: page index 3, y = 100.
Dim entryChapter2 As OutLine = New OutLine("Chapter 2: ******* ****", 1,
3, 100)

' Add 2nd entry to outline.
outline.Entry.Add(entryChapter2)

' Create 3rd entry: Level = 1; Location: page index 5, y = 200.
Dim entryChapter3 As OutLine = New OutLine("Chapter 3: **** **** ***", 1,
5, 200)

' Add 3rd entry to outline.
outline.Entry.Add(entryChapter3)

' Create 3rd entry's child entries.
' Create 1st child entry: Level = 2; Location: page index 9, y = 0.
Dim entryChapter31 As OutLine = New OutLine("# #### ###", 2, 9, 0)

' Connect this entry to the parent entry.
```

```vb
entryChapter31.SetParentREEntry(entryChapter3)

' Add 1st child of 3rd entry to outline.
outline.Entry.Add(entryChapter31)

' Create a child entry for the 1st child entry of the 3rd entry.
' Create entry: Level = 3; Location: page index 9, y = 500.
Dim entryChapter311 As OutLine = New OutLine("???? ???? ??? ???", 3, 9, 500)

' Connect this entry to the parent entry.
entryChapter311.SetParentREEntry(entryChapter31)

' Add 1st child of 3rd entry to outline.
outline.Entry.Add(entryChapter311)

' Create 4th entry: Level = 1; Location: page index 10, y = 0.
Dim entryChapter4 As OutLine = New OutLine("Chapter 4: ******* ** ** ****
***", 1, 10, 0)

' Add 4th entry to outline.
outline.Entry.Add(entryChapter4)

' Create 5th entry: Level = 1; Location: page index 20, y = 0.
Dim entryChapter5 As OutLine = New OutLine("Chapter 5: * ** **** **** ****",
1, 20, 0)

' Add 5th entry to outline.
outline.Entry.Add(entryChapter5)

' Update the new outline.
doc.SetOutline(outline)
doc.Save(outputFilePath)
```

Remove/Delete bookmarks

By using the RasterEdge.XDoc.PDF library, you can remove the bookmark label according to your own needs freely.

The following codes will show how to remove the specified bookmarks without limitation:

C#

```csharp
//Open one PDF file
String inputFilePath = "C:\\input.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
// Retrieve PDF document outline.
REOutline outline = doc.GetOutline();
//Remove the specified bookmark.
int removeIndex = 2;
if (0 < removeIndex && removeIndex < outline.Entry.Count )
    outline.Entry.RemoveAt(2);
//Update the modified bookmark.
doc.SetOutline(outline);
//Save the file to a new one.
doc.Save("C:\\modify_bookmark.pdf");
```

VB.NET

```vbnet
'Open one PDF file
Dim inputFilePath As String = "C:\\input.pdf"
Dim doc As PDFDocument = New PDFDocument(inputFilePath)
'Retrieve the PDF outline
Dim outline As REOutline = doc.GetOutline()
'Remove the specified bookmark.
Dim removeIndex As Integer = 0
If 0 < removeIndex And removeIndex < outline.Entry.Count Then
    outline.Entry.RemoveAt(removeIndex)
End If
'Update the bookmark labels
doc.SetOutline(outline)
'Save the modified file to a new one
doc.Save("C:\\modify_bookmark.pdf")
```

PDF Printing

Please refer to the following site:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-print/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-print/

If you need to print PDF file more methods, you can try other APIs in the class PDFPrinter.cs.

Print PDF to default printer

Print PDF to a specified printer

Print PDF to physical or virtual printer

Print PDF to XPS printer

Hide print dialog while printing PDF

Compression or Optimizing

Please refer to the following site:

http://www.rasteredge.com/how-to/csharp-imaging/pdf-compressing/
http://www.rasteredge.com/how-to/vb-net-imaging/pdf-compressing/

Reduce image size

Remove unused fonts

Delete unused fields