XDoc.PDF Developer Guide – Content Extraction Module

Table of Contents

Extract all images from a document

```
// open a document
String inputFilePath = Program.RootPath + "\\" + "3.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// extract all images in the document
List<PDFImage> allImages = PDFImageHandler.ExtractImages(doc);

// show information of these images
foreach (PDFImage image in allImages)
{
   Console.WriteLine("Image: page index = " + image.PageIndex);
   Console.WriteLine("    : X = " + image.Position.X + ", Y = " + image.Position.Y);
   Console.WriteLine("    : X = " + image.GetBoundary().X + ", Y = " + image.GetBoundary().Y);
   Console.WriteLine("    : Width = " + image.GetBoundary().Width);
   Console.WriteLine("    : Height = " + image.GetBoundary().Height);
}

// extract all images in the first page
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);
List<PDFImage> allImagesInPage = PDFImageHandler.ExtractImages(page);

// show information of these images
foreach (PDFImage image in allImagesInPage)
{
   Console.WriteLine("Image: page index = " + image.PageIndex);
   Console.WriteLine("    : X = " + image.Position.X + ", Y = " + image.Position.Y);
   Console.WriteLine("    : X = " + image.GetBoundary().X + ", Y = " + image.GetBoundary().Y);
   Console.WriteLine("    : Width = " + image.GetBoundary().Width);
   Console.WriteLine("    : Height = " + image.GetBoundary().Height);
}
```

Select an image in a PDF page

By position:

```
// open a document
String inputFilePath = Program.RootPath + "\\" + "3.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// get the first page
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);

// select image at the position (100F, 100F) in the page
PointF cursorPos = new PointF(100F, 100F);
PDFImage image = PDFImageHandler.SelectImage(page, cursorPos);
if (image == null)
{
   Console.WriteLine("No image has been found!");
}
else
{
   Console.WriteLine("Image: boundary = " + image.GetBoundary().ToString());
}
```

By region:

```
// open a document
String inputFilePath = Program.RootPath + "\\" + "3.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);

// get the first page
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);

// define the region (Rectangle [50F, 50F, 300F, 400F]) of the page
RectangleF region = new RectangleF(50F, 50F, 300F, 400F);

// get all images in the region in sequence (from bottom to top)
List<PDFImage> images = PDFImageHandler.SelectImages(page, region);

// select the top image in the region
PDFImage image1 = PDFImageHandler.SelectImage(page, region);

// select the bottom image in the region
int sequenceIndex = 0;
PDFImage image2 = PDFImageHandler.SelectImage(page, region, sequenceIndex);
```

Use text manager to retrieve text contents in a page

```
//  open a document
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
PDFDocument doc = new PDFDocument(inputFilePath);
//  get text manager from the document
PDFTextMgr textMgr = PDFTextHandler.ExportPDFTextManager(doc);

//  extract different text content from the first page
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);

//  get all characters in the page
List<PDFTextCharacter> allChars = textMgr.ExtractTextCharacter(page);
//  report characters
foreach (PDFTextCharacter obj in allChars)
{
   Console.WriteLine("Char: " + obj.GetChar() + "; Boundary: " + obj.GetBoundary().ToString());
}

//  get all words in the page
List<PDFTextWord> allWords = textMgr.ExtractTextWord(page);
//  report characters
foreach (PDFTextWord obj in allWords)
{
   Console.WriteLine("Word: " + obj.GetContent() + "; Boundary: " + obj.GetBoundary().ToString());
}

//  get all lines in the page
List<PDFTextLine> allLines = textMgr.ExtractTextLine(page);
//  report characters
foreach (PDFTextLine obj in allLines)
{
   Console.WriteLine("Line: " + obj.GetContent() + "; Boundary: " + obj.GetBoundary().ToString());
}
```

Select a text item in a PDF page

Select characters:

```
// open a document
String inputFilePath = Program.RootPath + "\\" + "2.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 (245F, 155F)
PointF cursor = new PointF(245F, 155F);
PDFTextCharacter aChar = textMgr.SelectChar(page, cursor);
if (aChar == null)
{
   Console.WriteLine("No character has been found.");
}
else
{
   Console.WriteLine("Value: " + aChar.GetChar() + "; Boundary: " + aChar.GetBoundary().ToString());
}

// select chars in the region (250F, 150F, 100F, 100F)
RectangleF region = new RectangleF(250F, 150F, 100F, 100F);
List<PDFTextCharacter> chars = textMgr.SelectChar(page, region);
foreach (PDFTextCharacter obj in chars)
{
   Console.WriteLine("Value: " + obj.GetChar() + "; Boundary: " + obj.GetBoundary().ToString());
}
```

Select a line:

```
// select a line at 150F from the top of the page
PDFTextLine aLine = textMgr.SelectLine(page, 150F);
if (aLine == null)
{
   Console.WriteLine("No character has been found.");
}
else
{
   Console.WriteLine("Line: " + aLine.GetContent());
}
```

Add a Bitmap image to a PDF page with the specified position

C#

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

//  load a sample image
Bitmap anImage = new Bitmap(Program.RootPath + "\\" + "1.png");

//  open the document
PDFDocument doc = new PDFDocument(inputFilePath);
//  get the first page
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 document
doc.Save(outputFilePath);
```

VB

Delete all images in the document

```
String inputFilePath = Program.RootPath + "\\" + "3.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

//  open the document
PDFDocument doc = new PDFDocument(inputFilePath);
//  extract all images from the document
List<PDFImage> allImages = PDFImageHandler.ExtractImages(doc);
//  delete all images from the document
foreach (PDFImage image in allImages)
{
   PDFImageHandler.DeleteImage(doc, image);
}

//  output the new document
doc.Save(outputFilePath);
```

Delete a character in the page

C#

```
// open a document
String inputFilePath = Program.RootPath + "\\" + "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 = Program.RootPath + "\\" + "output.pdf";
doc.Save(outputFilePath);
```

VB

```
' open a document
Dim inputFilePath As String = Program.RootPath + "\\" + "1.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 = Program.RootPath + "\\" + "output.pdf"
doc.Save(outputFilePath)
```

Delete characters in the page

```
//  open a document
String inputFilePath = Program.RootPath + "\\" + "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);

//  extract all characters in the page
List<PDFTextCharacter> chars = textMgr.ExtractTextCharacter(page);

int cnt = 0;
//  delete a character every 3 characters
foreach (PDFTextCharacter aChar in chars)
{
   if (cnt % 3 == 0)
   {
      textMgr.DeleteChar(aChar);
   }
   cnt++;
}

//  output the new document
String outputFilePath = Program.RootPath + "\\" + "output.pdf";
doc.Save(outputFilePath);
```

Add a single character to the page

```
// open a document
String inputFilePath = Program.RootPath + "\\" + "1.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 = Program.RootPath + "\\" + "output.pdf";
doc.Save(outputFilePath);
```

Add a string to the page

```
// open a document
String inputFilePath = Program.RootPath + "\\" + "1.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
textMgr.AddString(msg, font, pageIndex, cursor, fontColor);

// output the new document
String outputFilePath = Program.RootPath + "\\" + "output.pdf";
doc.Save(outputFilePath);
```

Copy an image from a document and paste to another position

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

PDFDocument doc = new PDFDocument(inputFilePath);

//  get the first page
int pageIndex = 0;
PDFPage page1 = (PDFPage)doc.GetPage(pageIndex);

//  select image at the position (480F, 550F) in the page
PointF cursorPos = new PointF(480F, 550F);
PDFImage image = PDFImageHandler.SelectImage(page1, cursorPos);

//  copy the image
Bitmap anImage = (Bitmap)image.Image.Clone();

//  get the second page
PDFPage page2 = (PDFPage)doc.GetPage(1);
//  set image position in the page: X = 100F, Y = 400F
PointF position = new PointF(100F, 400F);

//  add image to the page
PDFImageHandler.AddImage(page2, anImage, position);

//  output the new document
doc.Save(outputFilePath);
```

Redact all characters in a page region

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

// open document
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 3rd page
PDFPage page = (PDFPage)doc.GetPage(2);
// set redact region
RectangleF region = new RectangleF(100F, 100F, 300F, 300F);

// create redaction option
RedactionOptions options = new RedactionOptions();
options.AreaFillColor = Color.Black;

// process redaction
PDFTextHandler.RedactText(page, region, options);
// output file
doc.Save(outputFilePath);
```

Redact an image in the page

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

// open document
PDFDocument doc = new PDFDocument(inputFilePath);
// get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);

List<PDFImage> images = PDFImageHandler.ExtractImages(page);
if (images == null || images.Count == 0) return;

// create redaction option
RedactionOptions options = new RedactionOptions();
options.AreaFillColor = Color.LightGray;

// redact the image in the page
PDFImageHandler.RedactImage(page, images[0], options);
// output file
doc.Save(outputFilePath);
```

Redact the whole page

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

//  open document
PDFDocument doc = new PDFDocument(inputFilePath);
//  get the 1st page
PDFPage page = (PDFPage)doc.GetPage(0);

//  create redaction option
RedactionOptions options = new RedactionOptions();
options.AreaFillColor = Color.Black;

//  redact the whole page
page.Redact(options);
//  output file
doc.Save(outputFilePath);
```

Move an image in the page

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

//  open a document and select the page
PDFDocument doc = new PDFDocument(inputFilePath);
PDFPage page = (PDFPage)doc.GetPage(0);
//  extract all images in the page
List<PDFImage> images = PDFImageHandler.ExtractImages(page);
//  move the first image to position (0, 0) in the same page
PDFImageHandler.MoveImageTo(doc, images[0], 0, new PointF(0F, 0F));

//  output the new document
doc.Save(outputFilePath);
```

Resize an image in the page

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

//  open a document and select the page
PDFDocument doc = new PDFDocument(inputFilePath);
PDFPage page = (PDFPage)doc.GetPage(0);
//  extract all images in the page
List<PDFImage> images = PDFImageHandler.ExtractImages(page);
//  enlarge the first image with factor 1.2F
PDFImageHandler.ResizeImage(doc, images[0], 1.5F);

//  output the new document
doc.Save(outputFilePath);
```

Flip an image in the page

Horizontal Flip:

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "output1.pdf";

//  open a document and select the page
PDFDocument doc = new PDFDocument(inputFilePath);
PDFPage page = (PDFPage)doc.GetPage(0);
//  extract all images in the page
List<PDFImage> images = PDFImageHandler.ExtractImages(page);
//  horizontal flip the first image
PDFImageHandler.FlipImage(doc, images[0], FlipMode.FlipX);

//  output the new document
doc.Save(outputFilePath);
```

Vertical Flip:

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "output2.pdf";

//  open a document and select the page
PDFDocument doc = new PDFDocument(inputFilePath);
PDFPage page = (PDFPage)doc.GetPage(0);
//  extract all images in the page
List<PDFImage> images = PDFImageHandler.ExtractImages(page);
//  vertical flip the first image
PDFImageHandler.FlipImage(doc, images[0], FlipMode.FlipY);

//  output the new document
doc.Save(outputFilePath);
```

Change image's resolution

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "output2.pdf";

PDFDocument doc = new PDFDocument(inputFilePath);

PDFPage page = (PDFPage)doc.GetPage(0);

List<PDFImage> images = PDFImageHandler.ExtractImages(page);

PDFImageHandler.ReduceImageSize(doc, images[0], 2F);

doc.Save(outputFilePath);
```

Crop an image

```
String inputFilePath = Program.RootPath + "\\" + "2.pdf";
String outputFilePath = Program.RootPath + "\\" + "output2.pdf";

PDFDocument doc = new PDFDocument(inputFilePath);

PDFPage page = (PDFPage)doc.GetPage(0);

List<PDFImage> images = PDFImageHandler.ExtractImages(page);

PDFImageHandler.CropImage(doc, images[0], new Rectangle(0, 0, 50, 50));

doc.Save(outputFilePath);
```

Add a Bitmap image to a page with advanced settings

C#

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

//  load a sample image
Bitmap anImage = new Bitmap(Program.RootPath + "\\" + "1.png");

//  set image item option
PDFItemOptions ops = new PDFItemOptions();
//  set image position in the page: X = 100F, Y = 400F (in pixel, 96dpi)
ops.Position = new PointF(100F, 400F);
//  add image over all page contents
ops.Level = DisplayLevel.Over;
//  set image actual width and height in the page
ops.Width = 192;       //  2 inches in width
ops.Height = 96;       //  1 inch in height
//  set compression mode for the image
ops.Compression = PDFCompression.DCTDecode;
//  set image quality level (only available for compression mode DCT)
ops.JPEGImageQualityLevel = JPEGImageQualityLevel.Medium;

//  open the document
PDFDocument doc = new PDFDocument(inputFilePath);
int pageIndex = 0;

//  add image to the target page
PDFImageHandler.AddImage(doc, pageIndex, anImage, ops);

//  output the new document
doc.Save(outputFilePath);
```

VB

Redact page content with overlay text

Redact text in the page

C#

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

// open file and get the first page
PDFDocument doc = new PDFDocument(inputFilePath);
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);

// set the redact region
RectangleF redactRegion = new RectangleF(10F, 10F, 400F, 300F);

RedactionOptions ops = new RedactionOptions();
// set redect fill color: black
ops.AreaFillColor = Color.Black;
// enable overlay text
ops.EnableOverlayText = true;
// set overlay message
ops.OverlayText = @"Confidential";
// set font of the overlay text
ops.OverlayTextFont = new Font("Arial", 8F, FontStyle.Regular);
// set color of the overlay text
ops.OverlayTextColor = Color.Red;
// center alignment for the overlay message
ops.OverlayTextAlignment = OverlayTextAlignment.Center;
// repeat the message to fill the whole redect region
ops.IsRepeat = true;
// use the font size given above
ops.IsAutoSize = false;

// apply redaction
PDFTextHandler.RedactText(page, redactRegion, ops);

doc.Save(outputFilePath);
```

VB

Redact the whole page

C#

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

//  open file and get the first page
PDFDocument doc = new PDFDocument(inputFilePath);
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);

RedactionOptions ops = new RedactionOptions();
//  set redect fill color: black
ops.AreaFillColor = Color.Black;
//  enable overlay text
ops.EnableOverlayText = true;
//  set overlay message
ops.OverlayText = @"Confidential";
//  set font of the overlay text
ops.OverlayTextFont = new Font("Arial", 8F, FontStyle.Italic);
//  set color of the overlay text
ops.OverlayTextColor = Color.Red;
//  center alignment for the overlay message
ops.OverlayTextAlignment = OverlayTextAlignment.Center;
//  show overlay text once
ops.IsRepeat = false;
//  auto choose the font size of the text (font size in OverlayTextFont would be ignored if this flag is true)
ops.IsAutoSize = true;

//  apply redaction for the whole page
page.Redact(ops);

doc.Save(outputFilePath);
```

VB

Redact contents in the given area of a page

C#

```
String inputFilePath = Program.RootPath + "\\" + "1.pdf";
String outputFilePath = Program.RootPath + "\\" + "output.pdf";

//  open file and get the first page
PDFDocument doc = new PDFDocument(inputFilePath);
int pageIndex = 0;
PDFPage page = (PDFPage)doc.GetPage(pageIndex);

//  set a redact area start from point (200, 300) with size (200, 150)
//  all value in pixels (96 dpi)
RectangleF redactArea = new RectangleF(200, 300, 200, 150);

//  use default redact options
RedactionOptions ops = new RedactionOptions();

//  apply redaction for the whole page
page.Redact(redactArea, ops);

doc.Save(outputFilePath);
```

VB