HOWTO: Extract Text from an Office Document

Before 10.7, the only type of "extract text from document" that DotImage offered was with our PDF Text Extraction (PdfTextDocument class).

With the introduction of 10.7, our web components added a text search feature. Although direct text extraction from Office documents was not the primary target here, it turns out we can take advantage of our web controls to perform similar text extraction from Office documents, even in a Windows Forms or console application.

Licensing

In order to perform text extraction from Office documents, you need a license for both DotImage Document Imaging (our base SDK) and for Office addon.

We will be using our OfficeDecoder class in order to accomplish the text extraction.

Oddities

The actual classes we need for the text extraction "live in" our Atalasoft.dotImage.WebControls.dll, so even if you're using this technique away from the web, you will need to include this reference

Setting up the Project

In this example, we are going to make a Console application, so you will need to start with a basic console app.

The specific setup for the attached example project requires that you run this in a [STAThread] so in your program.cs you will need to add

```
[STAThread]
```

before the call to

```
static void Main(string[] args)
```

References

HOWTO: Extract Text from an Office Document

You will need to reference System.Windows.Forms to make use of the OpenFileDialog in this sample, but it is not required for use of the text extract - it's just convenient for giving you access to OpenFileDialog instead of having to ask you to type the file path in.

Additionally, you will need to reference

Atalasoft.dotImage.dll

Atalasoft.dotImage.Lib.dll

Atalasoft.dotImage.Office.dll

Atalasoft.dotImage.WebControls.dll

Atalasoft.Shared.dll

Non-Reference Dependencies

You will need to add the following files to your project (but they can't be added as references... they are found in

C:\Program Files (x86)\Atalasoft\DotImage 10.7\bin\PerceptiveDocumentFilters\intel-32\

ISYS11df.dll

ISYSreaders.dll

ISYSreadershd.dll

Perceptive.DocumentFilters.dll

You must go to the properties of each and set their Copy To Output Directory setting to "Always Copy" or "Copy if Newer"

The code

HOWTO: Extract Text from an Office Document

```
using System.Text;
using Atalasoft.Imaging;
using Atalasoft.Imaging.Codec;
using Atalasoft.Imaging.Codec.Office;
using System.IO;
using System.Diagnostics;
using Atalasoft.Imaging.WebControls.Text;
using Atalasoft.Imaging.Text;
using System.Windows.Forms;

namespace OfficeDecoder_TextExtractionExample
{
    class Program
    {
        [STAThread]
        static void Main(string[] args)
        {
            // Critical - this adds support for the Office file types..
without it, the extraction won't work
            RegisteredDecoders.Decoders.Add(new OfficeDecoder() { Resolution =
200 });

            Console.WriteLine("OfficeDecoder_TextExtractionExample
Starting...");
            string imgPath = GetWorkingDir();
            string inFile = imgPath + "target.docx";
```

```
    //dlg.FileName = inFile;
    dlg.InitialDirectory = imgPath;
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        inFile = dlg.FileName;
    }
}

Console.WriteLine("  inFile: " + inFile);
string outFile = inFile + ".out.txt";
Console.WriteLine("  outFile: " + outFile);

Console.WriteLine("BEGIN Processing");

// This is where we will store the text output
List<string> output = new List<string>();
```

```
                            // for documents that have comlicated structure,

                            //i.e. consist from the isolated pieces of text,
or table structure

                            // it's possible to configure nearby text blocks
are combined into text

                            // segments(text containers that provide

                            // selection isolated from other document content)

                            extractor.RegionDetection =
TextRegionDetectionMode.LineDetection;

                            // each block boundaries inflated to one average
character width and two average character height

                            // and all intersecting blocks are combined into
single segment.

                            // Having vertical ratio bigger then horizontal
behaves better on column-layout documents.

                            //extractor.BlockDetectionDistance = new
System.Drawing.SizeF(1, 2);

                            int pageCount = extractor.TextDocument.PageCount;

                            Console.WriteLine("Document open.. toal number of
pages: " + pageCount.ToString());
```

```
                        {
                            foreach (Region region in page.Regions)
                            {
                                foreach (Line line in region.Lines)
                                {
                                    string lineText = "";
                                    foreach (Word word in line.Words)
                                    {
                                        lineText += word.Text + " ";
                                    }
                                    output.Add(lineText);
                                    Console.WriteLine(lineText);
                                }
                            }
                        }
                    }
                }
                catch (ImageReadException imagingException)
                {
                    Console.WriteLine("Text extraction: image type is not
recognized. {0}", imagingException);
                }
            }
```

```
            {
                Console.WriteLine("\n\nsaving output to " + outFile);
                if (File.Exists(outFile))
                {
                    File.Delete(outFile);
                }

                using (TextWriter tw = new StreamWriter(outFile))
                {
                    foreach (String line in output)
                    {
                        tw.WriteLine(line);
                    }
                    tw.Close();
                }
            }
            else
            {
                Console.WriteLine("\n\nNo text extracted.. nothing to save");
            }

            Console.WriteLine("\nEND Processing");

            Console.WriteLine("OfficeDecoder_TextExtractionExample finished...
press RETURN to exit");
            Console.ReadLine();
        }
```

```
        /// Convenience method to get the root directory of the project -
really only useful for debugging
        /// </summary>
        /// <returns></returns>
        private static string GetWorkingDir()
        {
            string cwd = System.IO.Directory.GetCurrentDirectory();
            //Console.WriteLine("cwd is '{0}'", cwd);

            if (cwd.EndsWith("\\bin\\Debug"))
            {
                cwd = cwd.Replace("\\bin\\Debug", "\\..\\");
                //Console.WriteLine("updated cwd is '{0}'", cwd);
            }
            return cwd;
        }
    }
}
```

Original Article:

Q10447 - HOWTO: Extract Text from Office Document

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50054/HOWTO-Extract-Text-from-an-Office-Do...