textextractor progress indication in C# and ByteScout PDF Extractor SDK

How to code textextractor progress indication in C#: How-To tutorial

Here you may find thousands pre-made source code pieces for easy implementation in your own programming C# projects. Textextractor progress indication in C# can be implemented with ByteScout PDF Extractor SDK. ByteScout PDF Extractor SDK is the SDK is designed to help developers with pdf tables and pdf data extraction from unstructured documents like pdf, tiff, scans, images, scanned and electronic forms. The library is powered by OCR, computer vision and AI to provide unique functionality like table detection, automatic table structure extraction, data restoration, data restructuring and reconstruction. Supports PDF, TIFF, PNG, JPG images as input and can output CSV, XML, JSON formatted data. Includes full set of utilities like pdf splitter, pdf merger, searchable pdf maker and other utilities.

This rich sample source code in C# for ByteScout PDF Extractor SDK includes the number of functions and options you should do calling the API to implement textextractor progress indication. This C# sample code should be copied and pasted into your application's code editor. Then just compile and run it to see how it works. Enjoy writing a code with ready-to-use sample C# codes to add textextractor progress indication functions using ByteScout PDF Extractor SDK in C#.

On our website you may get trial version of ByteScout PDF Extractor SDK for free. Source code samples are included to help you with your C# application.

C# - Program.cs

```
using Bytescout.PDFExtractor;
using System;

namespace TextExtractorProgressChangedEvent
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                //Read all file content...
                using (TextExtractor extractor = new TextExtractor())
                {
                    // Load document
                    extractor.LoadDocumentFromFile("sample.png");

                    // Extractor Progress event
                    Console.WriteLine("Text Extraction in progress: \n");
                    extractor.ProgressChanged += Extractor_ProgressChanged;
                    // Set option to repair text
                    extractor.OCRMode =
OCRMode.TextFromImagesAndVectorsAndRepairedFonts;

                    // Enable Optical Character Recognition (OCR)
                    // in .Auto mode (SDK automatically checks if needs to use OCR or
not)
                    extractor.OCRMode = OCRMode.Auto;

                    // Set the location of OCR language data files
                    extractor.OCRLanguageDataFolder = @"c:\Program Files\Bytescout
PDF Extractor SDK\ocrdata\";

                    // Set OCR language
                    extractor.OCRLanguage = "eng"; // "eng" for english, "deu" for
German, "fra" for French, "spa" for Spanish etc - according to files in "ocrdata"
folder
                    // Find more language files at
https://github.com/bytescout/ocrdata

                    // Set PDF document rendering resolution
                    extractor.OCRResolution = 300;

                    //Read all text
                    var allExtractedText = extractor.GetText();
                    Console.WriteLine("\n\nExtracted Text:\n\n{0}",
allExtractedText);
                }


            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }

                        Console.WriteLine("Press enter key to exit...");
            Console.ReadLine();
        }

        ///

        /// Handle progress change event
        ///

        private static void Extractor_ProgressChanged(object sender, OngoingOperation
ongoingOperation, double progress, ref bool cancel)
        {
            drawTextProgressBar(Convert.ToInt32(progress), 100);
        }

        ///

        /// Display progress bar
        ///

        private static void drawTextProgressBar(int progress, int total)
        {
            //draw empty progress bar
            Console.CursorLeft = 0;
            Console.Write("["); //start
            Console.CursorLeft = 32;
            Console.Write("]"); //end
            Console.CursorLeft = 1;
            float onechunk = 30.0f / total;

            //draw filled part
            int position = 1;
            for (int i = 0; i < onechunk * progress; i++)
            {
                Console.BackgroundColor = ConsoleColor.Green;
                Console.CursorLeft = position++;
                Console.Write(" ");
            }

            //draw unfilled part
            for (int i = position; i <= 31; i++)
            {
                Console.BackgroundColor = ConsoleColor.Gray;
                Console.CursorLeft = position++;
                Console.Write(" ");
            }

            //draw totals
            Console.CursorLeft = 35;
            Console.BackgroundColor = ConsoleColor.Black;
            Console.Write(progress.ToString() + " of " + total.ToString() + "    ");
//blanks at the end remove any excess
        }
    }
}
```
