www.bytescout.com

How to make large PDF document searchable using parallel processing in C# using ByteScout PDF Suite

Learn to code in C# to make large PDF document searchable using parallel processing with this step-by-step tutorial

These source code samples are assembled by their programming language and functions they apply. ByteScout PDF Suite is the set that includes 6 SDK products to work with PDF from generating rich PDF reports to extracting data from PDF documents and converting them to HTML. This bundle includes PDF (Generator) SDK, PDF Renderer SDK, PDF Extractor SDK, PDF to HTML SDK, PDF Viewer SDK and PDF Generator SDK for Javascript. It can make large PDF document searchable using parallel processing in C#.

The SDK samples given below describe how to quickly make your application do make large PDF document searchable using parallel processing in C# with the help of ByteScout PDF Suite. Follow the instructions from scratch to work and copy the C# code. Check C# sample code samples to see if they respond to your needs and requirements for the project.

You can download free trial version of ByteScout PDF Suite from our website with this and other source code samples for C#.

FOR MORE INFORMATION AND FREE TRIAL:

Download Free Trial SDK (on-premise version)

Read more about ByteScout PDF Suite

Explore API Documentation

Get Free Training for ByteScout PDF Suite

Get Free API key for Web API

visit www.ByteScout.com

Source Code Files:

MultithreadProcessing.sln

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 15
VisualStudioVersion = 15.0.26730.10
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "MultithreadProcessing", "Multithre
EndProject
Global
	GlobalSection(SolutionConfigurationPlatforms) = preSolution
		Debug|Any CPU = Debug|Any CPU
		Release|Any CPU = Release|Any CPU
	EndGlobalSection
	GlobalSection(ProjectConfigurationPlatforms) = postSolution
		{0B102DA4-C143-481D-A076-1F56E3CB1CF5}.Debug|Any CPU.ActiveCfg = Debug
		{0B102DA4-C143-481D-A076-1F56E3CB1CF5}.Debug|Any CPU.Build.0 = Debug|A
		{0B102DA4-C143-481D-A076-1F56E3CB1CF5}.Release|Any CPU.ActiveCfg = Rele
		{0B102DA4-C143-481D-A076-1F56E3CB1CF5}.Release|Any CPU.Build.0 = Releas
	EndGlobalSection
	GlobalSection(SolutionProperties) = preSolution
		HideSolutionNode = FALSE
	EndGlobalSection
	GlobalSection(ExtensibilityGlobals) = postSolution
		SolutionGuid = {50466307-7059-438B-8545-42FDA71BC1A6}
	EndGlobalSection
EndGlobal
```

Program.cs

```
using System;
using System.Diagnostics;
using System.IO;
using System.Threading;
using Bytescout.PDFExtractor;

namespace MultithreadProcessing
{
	class Program
	{
	    // Limit to 4 threads in queue.
	    // Set this value to number of your processor cores for max performance.
	    private static readonly Semaphore ThreadLimiter = new Semaphore(4, 4);

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

```
                const string inputFile = "sample.pdf";
                const string resultFile = "result.pdf";

                int pageCount;

                // Get document page count
                using (var infoExtractor = new InfoExtractor("demo", "demo"))
                {
                        infoExtractor.LoadDocumentFromFile(inputFile);
                        pageCount = infoExtractor.GetPageCount();
                }

                // Process the document by 10-page pieces

                int numberOfThreads = pageCount / 10;
                if (pageCount - numberOfThreads * 10 > 0)
                        numberOfThreads += 1;

                WaitHandle[] doneEvents = new WaitHandle[numberOfThreads];
                Stopwatch stopwatch = Stopwatch.StartNew();
                int startPage, endPage;
                string[] pieces = new string[numberOfThreads];

                for (int i = 0; i < numberOfThreads; i++)
                {
                    // Wait for the queue
                    ThreadLimiter.WaitOne();

        doneEvents[i] = new ManualResetEvent(false);
                        startPage = i * 10;
                        endPage = Math.Min(pageCount - 1, (i + 1) * 10 - 1);

                        pieces[i] = string.Format("temp-{0}-{1}.pdf", startPage
                        ThreadPool.QueueUserWorkItem(new WaitCallback(ThreadPro
                                new object[] { i, doneEvents[i], inputFile, pie
                }

                // Wait for all threads
                WaitHandle.WaitAll(doneEvents);

                // Merge pieces
                using (DocumentMerger merger = new DocumentMerger("demo", "demo
                        merger.Merge(pieces, resultFile);

                // Delete temp files
                foreach (string tempFile in pieces)
                        File.Delete(tempFile);

                Console.WriteLine("All done in {0}.", stopwatch.Elapsed);
                Console.WriteLine();

                Console.WriteLine("Press any key to exit...");
                Console.ReadKey();
        }

        private static void ThreadProc(object stateInfo)
        {
                int threadIndex = (int) ((object[]) stateInfo)[0];
                ManualResetEvent doneEvent = (ManualResetEvent) ((object[]) sta
                string inputFile = (string) ((object[]) stateInfo)[2];
```

```
                    string outputFile = (string)((object[])stateInfo)[3];
                    int startPage = (int)((object[])stateInfo)[4];
                    int endPage = (int)((object[])stateInfo)[5];

            try
                    {
                        Console.WriteLine("Thread #{0} started with the page range fro

                        Stopwatch stopwatch = Stopwatch.StartNew();

                        // Extract a piece of document
                        string chunk = string.Format("temp-{0}-{1}", startPage, endPag
                        using (DocumentSplitter splitter = new DocumentSplitter("demo"
                            splitter.ExtractPageRange(inputFile, chunk, startPage + 1,

                        // Process the piece
                        using (SearchablePDFMaker searchablePdfMaker = new SearchablePD
                        {
                            searchablePdfMaker.OCRDetectPageRotation = true;
                            searchablePdfMaker.OCRLanguageDataFolder = @"C:\Program Fi
                            searchablePdfMaker.LoadDocumentFromFile(chunk);

                            // 300 DPI resolution is recommended.
                            // Using of higher values will slow down the processing bu
                            searchablePdfMaker.OCRResolution = 300;

                            searchablePdfMaker.MakePDFSearchable(outputFile);
                        }

                        File.Delete(chunk);

                        Console.WriteLine("Thread #{0} finished in {1}.", threadIndex,
                    }
            finally
            {
                // Signal the thread is finished
                doneEvent.Set();

                // Release semaphore
                ThreadLimiter.Release();
            }
        }
        }
}
```

VIDEO

https://www.youtube.com/watch?v=NEwNs2b9YN8

ON-PREMISE OFFLINE SDK

60 Day Free Trial or Visit ByteScout PDF Suite Home Page
Explore ByteScout PDF Suite Documentation
Explore Samples
Sign Up for ByteScout PDF Suite Online Training

ON-DEMAND REST WEB API

Get Your API Key
Explore Web API Docs
Explore Web API Samples

visit www.ByteScout.com

visit www.PDF.co