www.bytescout.com

How to parse uploaded file asynchronously for document parser API in C# with ByteScout Cloud API Server

Learn to write code parse uploaded file asynchronously for document parser API in C#: Simple How To Tutorial

This page displays the step-by-step instructions and algorithm of how to parse uploaded file asynchronously and how to apply it in your application. ByteScout Cloud API Server was designed to assist document parser API in C#. ByteScout Cloud API Server is API server that is ready to use and can be installed and deployed in less than 30 minutes on your own Windows server or server in a cloud. It can save data and files on your local server-based file storage or in Amazon AWS S3 storage. Data is processed solely on the API server and is powered by ByteScout engine, no cloud services or Internet connection is required for data processing..

Use the code displayed below in your application to save a lot of time on writing and testing code. This C# sample code can be used by copying and pasting into your project. Once done,just compile your project and click Run. Enjoy writing a code with ready-to-use sample C# codes to add document parser API functions using ByteScout Cloud API Server in C#.

ByteScout Cloud API Server - free trial version is available on our website. Also, there are other code samples to help you with your C# application included into trial version.

FOR MORE INFORMATION AND FREE TRIAL:

Download Free Trial SDK (on-premise version)

Read more about ByteScout Cloud API Server

Explore API Documentation

Get Free Training for ByteScout Cloud API Server

Get Free API key for Web API

visit www.ByteScout.com

Source Code Files:

ByteScoutWebApiExample.csproj

```
<?xml version="1.0" encoding="utf-8"?>
<Project ToolsVersion="12.0" DefaultTargets="Build" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
  <Import Project="$(MSBuildExtensionsPath)\$(MSBuildToolsVersion)\Microsoft.Common.props" Condition="Exists('$
  <PropertyGroup>
    <Configuration Condition=" '$(Configuration)' == '' ">Debug</Configuration>
    <Platform Condition=" '$(Platform)' == '' ">AnyCPU</Platform>
    <ProjectGuid>{1E1C2C34-017E-4605-AE2B-55EA3313BE51}</ProjectGuid>
    <OutputType>Exe</OutputType>
    <RootNamespace>ByteScoutWebApiExample</RootNamespace>
    <AssemblyName>ByteScoutWebApiExample</AssemblyName>
    <TargetFrameworkVersion>v4.0</TargetFrameworkVersion>
    <FileAlignment>512</FileAlignment>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugSymbols>true</DebugSymbols>
    <DebugType>full</DebugType>
    <Optimize>false</Optimize>
    <OutputPath>bin\Debug\</OutputPath>
    <DefineConstants>DEBUG;TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
    <PlatformTarget>AnyCPU</PlatformTarget>
    <DebugType>pdbonly</DebugType>
    <Optimize>true</Optimize>
    <OutputPath>bin\Release\</OutputPath>
    <DefineConstants>TRACE</DefineConstants>
    <ErrorReport>prompt</ErrorReport>
    <WarningLevel>4</WarningLevel>
  </PropertyGroup>
  <ItemGroup>
    <Reference Include="Newtonsoft.Json, Version=10.0.0.0, Culture=neutral, PublicKeyToken=30ad4fe6b2a6aeed, pr
      <HintPath>packages\Newtonsoft.Json.10.0.3\lib\net40\Newtonsoft.Json.dll</HintPath>
      <Private>True</Private>
    </Reference>
    <Reference Include="System" />
    <Reference Include="System.Core" />
    <Reference Include="System.Xml.Linq" />
    <Reference Include="System.Data" />
    <Reference Include="System.Xml" />
  </ItemGroup>
  <ItemGroup>
    <Compile Include="Program.cs" />
  </ItemGroup>
  <ItemGroup>
    <None Include="MultiPageTable-template1.yml">
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
    <None Include="MultiPageTable.pdf">
      <CopyToOutputDirectory>Always</CopyToOutputDirectory>
    </None>
    <None Include="packages.config" />
  </ItemGroup>
  <Import Project="$(MSBuildToolsPath)\Microsoft.CSharp.targets" />
  <!-- To modify your build process, add your task inside one of the targets below and uncomment it.
       Other similar extension points exist, see Microsoft.Common.targets.
  <Target Name="BeforeBuild">
  </Target>
  <Target Name="AfterBuild">
  </Target>
  -->
</Project>
```

ByteScoutWebApiExample.sln

```
Microsoft Visual Studio Solution File, Format Version 12.00
# Visual Studio 2013
VisualStudioVersion = 12.0.40629.0
MinimumVisualStudioVersion = 10.0.40219.1
Project("{FAE04EC0-301F-11D3-BF4B-00C04F79EFBC}") = "ByteScoutWebApiExample", "ByteScoutWebApiExamp
EndProject
Global
	GlobalSection(SolutionConfigurationPlatforms) = preSolution
		Debug|Any CPU = Debug|Any CPU
		Release|Any CPU = Release|Any CPU
	EndGlobalSection
	GlobalSection(ProjectConfigurationPlatforms) = postSolution
		{1E1C2C34-017E-4605-AE2B-55EA3313BE51}.Debug|Any CPU.ActiveCfg = Debug|Any CPU
		{1E1C2C34-017E-4605-AE2B-55EA3313BE51}.Debug|Any CPU.Build.0 = Debug|Any CPU
		{1E1C2C34-017E-4605-AE2B-55EA3313BE51}.Release|Any CPU.ActiveCfg = Release|Any CPU
		{1E1C2C34-017E-4605-AE2B-55EA3313BE51}.Release|Any CPU.Build.0 = Release|Any CPU
	EndGlobalSection
	GlobalSection(SolutionProperties) = preSolution
		HideSolutionNode = FALSE
	EndGlobalSection
EndGlobal
```

MultiPageTable-template1.yml

```
---
# Template that demonstrates parsing of multi-page table using only
# regular expressions for the table start, end, and rows.
# If regular expression cannot be written for every table row (for example,
# if the table contains empty cells), try the second method demonstrated
# in `MultiPageTable-template2.yml` template.
templateVersion: 2
templatePriority: 0
sourceId: Multipage Table Test
detectionRules:
  keywords:
  - Sample document with multi-page table
fields:
  total:
    expression: TOTAL {{DECIMAL}}
tables:
- name: table1
  start:
    # regular expression to find the table start in document
    expression: Item\s+Description\s+Price\s+Qty\s+Extended Price
  end:
    # regular expression to find the table end in document
    expression: TOTAL\s+\d+\.\d\d
  row:
    # regular expression to find table rows
    expression: '^\s*(?<itemNo>\d+)\s+(?<description>.+?)\s+(?<price>\d+\.\d\d)\s+(?<qty>\d+)\s+(?<extPrice>\d+\.\d\d
  columns:
  - name: itemNo
    type: integer
  - name: description
    type: string
  - name: price
```

```
    type: decimal
  - name: qty
    type: integer
  - name: extPrice
    type: decimal
  multipage: true
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Collections.Specialized;
using System.IO;
using System.Net;
using System.Text;
using System.Threading;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;

// Please NOTE: In this sample we're assuming Cloud Api Server is hosted at "https://localhost".
// If it's not then please replace this with with your hosting url.

// Cloud API asynchronous "Document Parser" job example.
// Allows to avoid timeout errors when processing huge or scanned PDF documents.

namespace ByteScoutWebApiExample
{
	class Program
	{
		// Source PDF file
		const string SourceFile = @".\MultiPageTable.pdf";

		// PDF document password. Leave empty for unprotected documents.
		const string Password = "";

		// Destination TXT file name
		const string DestinationFile = @".\result.json";

	// (!) Make asynchronous job
	const bool Async = true;


		static void Main(string[] args)
		{
			// Template text. Use Document Parser SDK (https://bytescout.com/products/developer/documentparse
			// to create templates.
			// Read template from file:
			String templateText = File.ReadAllText(@".\MultiPageTable-template1.yml");

			// Create standard .NET web client instance
			WebClient webClient = new WebClient();

			// 1. RETRIEVE THE PRESIGNED URL TO UPLOAD THE FILE.
			// * If you already have a direct file URL, skip to the step 3.

			// Prepare URL for `Get Presigned URL` API call
			string query = Uri.EscapeUriString(string.Format(
				"https://localhost/file/upload/get-presigned-url?contenttype=application/octet-stream&name={0}",
				Path.GetFileName(SourceFile)));

			try
			{
				// Execute request
				string response = webClient.DownloadString(query);

				// Parse JSON response
				JObject json = JObject.Parse(response);
```

```
            if (json["error"].ToObject<bool>() == false)
            {
                // Get URL to use for the file upload
                string uploadUrl = json["presignedUrl"].ToString();
                string uploadedFileUrl = json["url"].ToString();

                // 2. UPLOAD THE FILE TO CLOUD.

                webClient.Headers.Add("content-type", "application/octet-stream");
                webClient.UploadFile(uploadUrl, "PUT", SourceFile); // You can use UploadData() instead if y
                webClient.Headers.Remove("content-type");

                // 3. PARSE UPLOADED PDF DOCUMENT

// URL for `Document Parser` API call
query = Uri.EscapeUriString(string.Format(
    "https://localhost/pdf/documentparser?url={0}&async={1}",
    uploadedFileUrl,
    Async));

Dictionary<string, string> requestBody = new Dictionary<string, string>();
requestBody.Add("template", templateText);

// Execute request
response = webClient.UploadString(query, "POST", JsonConvert.SerializeObject(requestBody));

// Parse JSON response
json = JObject.Parse(response);

if (json["error"].ToObject<bool>() == false)
{
    // Asynchronous job ID
    string jobId = json["jobId"].ToString();
    // Get URL of generated JSON file
    string resultFileUrl = json["url"].ToString();

    // Check the job status in a loop.
    // If you don't want to pause the main thread you can rework the code
    // to use a separate thread for the status checking and completion.
    do
    {
        string status = CheckJobStatus(webClient, jobId); // Possible statuses: "working", "failed", "aborted",

        // Display timestamp and status (for demo purposes)
        Console.WriteLine(DateTime.Now.ToLongTimeString() + ": " + status);

        if (status == "success")
        {
            // Download JSON file
            webClient.DownloadFile(resultFileUrl, DestinationFile);

            Console.WriteLine("Generated JSON file saved as \"{0}\" file.", DestinationFile);
            break;
        }
        else if (status == "working")
        {
            // Pause for a few seconds
            Thread.Sleep(3000);
        }
        else
        {
            Console.WriteLine(status);
            break;
        }
    }
    while (true);
}
else
{
    Console.WriteLine(json["message"].ToString());
}
            }
            else
            {
                Console.WriteLine(json["message"].ToString());
            }
    }
```

```
                catch (WebException e)
                {
                        Console.WriteLine(e.ToString());
                }

                webClient.Dispose();


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

        static string CheckJobStatus(WebClient webClient, string jobId)
        {
            string url = "https://localhost/job/check?jobid=" + jobId;

            string response = webClient.DownloadString(url);
            JObject json = JObject.Parse(response);

            return Convert.ToString(json["status"]);
        }
        }
}
```

packages.config

```
<?xml version="1.0" encoding="utf-8"?>
<packages>
  <package id="Newtonsoft.Json" version="10.0.3" targetFramework="net40" />
</packages>
```

VIDEO

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

ON-PREMISE OFFLINE SDK

60 Day Free Trial or Visit ByteScout Cloud API Server Home Page
Explore ByteScout Cloud API Server Documentation
Explore Samples
Sign Up for ByteScout Cloud API Server 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