2x speed by parallel generation in VB.NET and ByteScout Image To Video SDK

How to code 2x speed by parallel generation in VB.NET: How-To tutorial

Source code documentation samples provide quick and easy way to add a required functionality into your application. ByteScout Image To Video SDK helps with 2x speed by parallel generation in VB.NET. ByteScout Image To Video SDK is the library for conversion of images into video slideshow. Provides built-in support for more than hundred of 2D and 3D transition effects. Output formats supported are AVI,WMV and WEBM video files. You may adjust output video size, quality, framerate and add audio.

The SDK samples like this one below explain how to quickly make your application do 2x speed by parallel generation in VB.NET with the help of ByteScout Image To Video SDK. In order to implement this functionality, you should copy and paste code below into your app using code editor. Then compile and run your application. Enjoy writing a code with ready-to-use sample VB.NET codes to add 2x speed by parallel generation functions using ByteScout Image To Video SDK in VB.NET.

Trial version can be downloaded from our website. Source code samples for VB.NET and documentation are included.

FOR MORE INFORMATION AND FREE TRIAL:

Download Free Trial SDK (on-premise version)

Read more about ByteScout Image To Video SDK

Explore API Documentation

Get Free Training for ByteScout Image To Video SDK

Get Free API key for Web API

visit www.ByteScout.com

Source Code Files:

Program.vb

```
' You can speed up the conversion using the following technique:
' 1) Generate video parts in parallel threads;
' 2) Combine these parts into final video.
' Let us say you have 20 slides.
' Then you can run a thread to convert 1-10 slides and another one to convert 11-20 sl
' Finally combine these parts into a single one using .JoinWMVFiles(part1, part2, outpu
' or .JoingAVIFiles(part1, part2, output) functions.

Imports System.Diagnostics
Imports System.Runtime.InteropServices
Imports System.Threading
Imports BytescoutImageToVideo

Class ThreadData
        Public InputFiles As String()
        Public OutputFile As String

    Public Sub New(ByVal inputFiles As String(), ByVal outputFile As String)
        Me.InputFiles = inputFiles
        Me.OutputFile = outputFile
    End Sub
End Class

Class Program
    Private Shared NumBusy As Integer
    Private Shared DoneEvent As ManualResetEvent

    Friend Shared Sub Main(ByVal args As String())
        DoneEvent = New ManualResetEvent(False)
        NumBusy = 2

        ' variable to store video file extension
        Dim videoFileExt As String

        ' are we using WMV or AVI video format for output
        Dim isWMV As Boolean = False

        ' if is WMV then use .wmv extension
        If isWMV Then
            videoFileExt = ".wmv"
        Else
            ' else use AVI
            videoFileExt = ".avi"
        End If

        ' Start two conversion threads
        Console.WriteLine("Start first thread...")
        Dim threadData1 As New ThreadData(New String() {"slide1.jpg", "slide2.jpg", "s
        ThreadPool.QueueUserWorkItem(AddressOf DoWork, threadData1)

        Console.WriteLine("Start second thread...")
        Dim threadData2 As New ThreadData(New String() {"slide4.jpg", "slide5.jpg", "s
        ThreadPool.QueueUserWorkItem(AddressOf DoWork, threadData2)

        ' Wait for both threads finished
        DoneEvent.WaitOne()

        ' Join generates parts
        Console.WriteLine("Join parts into the final video file...")
        Dim converter As New ImageToVideo()

        If isWMV Then
            converter.JoinWMVFiles("Part1" & videoFileExt, "Part2" & videoFileExt, "Res
        Else
            converter.JoinAVIFiles("Part1" & videoFileExt, "Part2" & videoFileExt, "Res
        End If

        ' Open the output video file in default media player
        Process.Start("Result" & videoFileExt)

        Console.WriteLine("Done. Press any key to continue...")
        Console.ReadKey()
    End Sub

    Private Shared Sub DoWork(ByVal data As Object)

        Dim threadData As ThreadData = DirectCast(data, ThreadData)

        Try
            ' Create BytescoutImageToVideoLib.ImageToVideo object instance
            Dim converter As New ImageToVideo()

            ' Activate the component
            converter.RegistrationName = "demo"
            converter.RegistrationKey = "demo"

            ' Add slides
            For Each file As String In threadData.InputFiles
                Dim slide As Slide = converter.AddImageFromFileName(file)
                slide.Duration = 3000 ' 3000ms = 3s
                slide.Effect = SlideEffectType.seEaseIn
            Next

            ' Set output video size
            converter.OutputWidth = 640
            converter.OutputHeight = 480

            ' Set output video file name
            converter.OutputVideoFileName = threadData.OutputFile

            ' Run the conversion
            converter.RunAndWait()

            ' Release resources
            Marshal.ReleaseComObject(converter)

            Console.WriteLine("Thread finished.")

        Catch ex As Exception
            Console.WriteLine(ex.Message)
        End Try

        If Interlocked.Decrement(NumBusy) = 0 Then
            DoneEvent.Set()
        End If
    End Sub
End Class
```

VIDEO

https://www.youtube.com/watch?v=FzPgeGFL8YA

ON-PREMISE OFFLINE SDK

60 Day Free Trial or Visit ByteScout Image To Video SDK Home Page
Explore ByteScout Image To Video SDK Documentation
Explore Samples
Sign Up for ByteScout Image To Video SDK 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