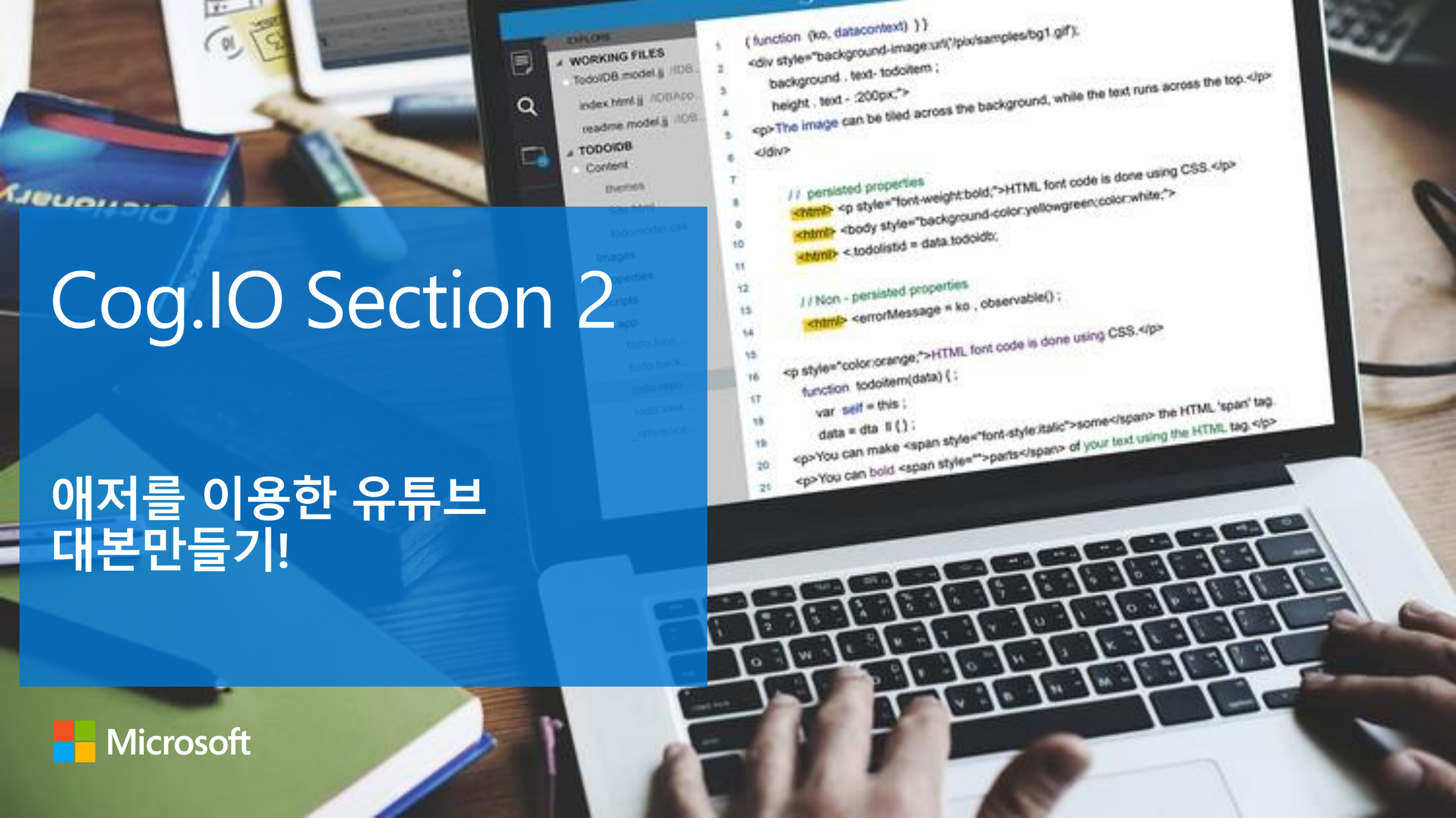


Cog.IO Section 2

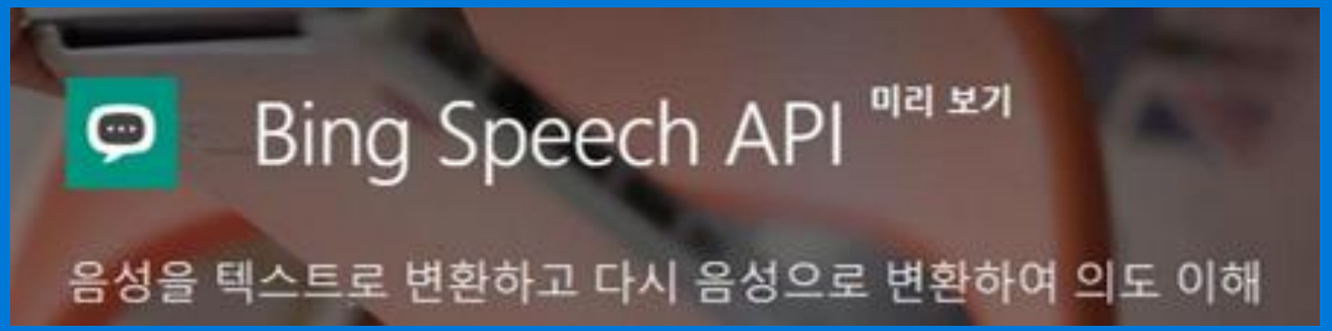
애저를 이용한 유튜브 대본만들기!

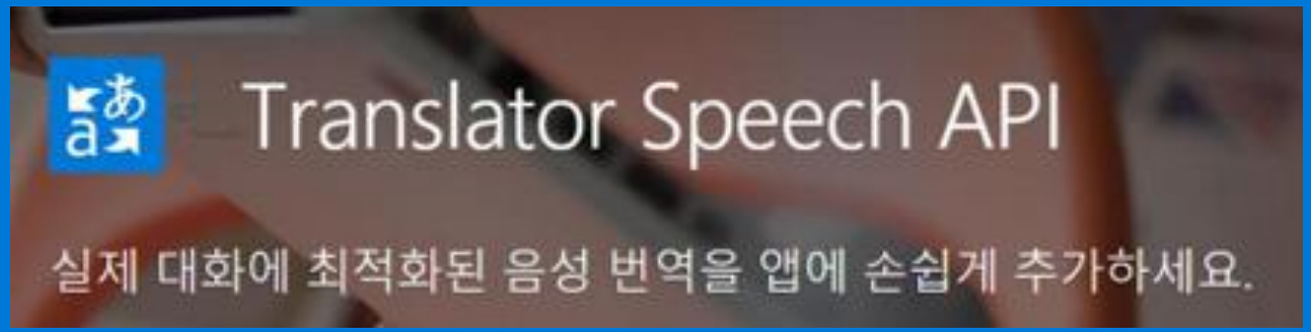
Microsoft

INTRO

INTRO

YouTube





INTRO

YoutubeExtractor

- Library for .NET
- C#
- 비디오 다운로드
- 오디오 트랙 추출

INTRO

Bing Speech API

- 음성 인식 파일의 음성 오디오를 실시간으로 텍스트 변환
- 음성 의도 인식 음성 오디오를 작업을 유도하는 의도로 변환
- 텍스트 음성 변환 텍스트를 음성 오디오로 변환

INTRO

Translator Speech API

- 클라우드 기반 자동 번역 서비스
- 9개 언어 번역
- 실제 대화의 번역에 최적화된 기술 사용

Why this program?

Why this program?

Why this program?

How?


```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Threading;
using System.Linq;
using System.Windows.Forms;
using YoutubeExtractor;
using System.Net;
using System.Net.Http;
using JDP;
using MediaToolkit.Model;
using MediaToolkit.Options;
using MediaToolkit;
using NAudio.Wave;
using WMPLib;
using Microsoft.CognitiveServices.SpeechRecognition;
using System.IO;
using System.Text;
using System.Web;
using System.Runtime.Serialization.Json;
using System.Threading.Tasks;
```

Show Demo

Extract Youtube

YoutubeExtractor

Donate

Flattr

Overview

YoutubeExtractor is a library for .NET, written in C#, that allows to download videos from YouTube and/or extract their audio track (audio extraction currently only for flash videos).

Target platforms

- .NET Framework 3.5 and higher
- Windows Phone 8
- WinRT
- Xamarin.Android
- Xamarin.iOS

Note that Windows Phone 8, WinRT, Xamarin.Android and Xamarin.iOS only support the extraction of the download URLs

Extract Youtube

```
// Our test youtube link
string link = "insert youtube link";

/*
 * Get the available video formats.
 * We'll work with them in the video and audio download examples.
 */
IEnumerable<VideoInfo> videoInfos = DownloadUrlResolver.GetDownloadUrls(link);
```

Extract Youtube

```
/*
 * Select the first .mp4 video with 360p resolution
 */
VideoInfo video = videoInfos
    .First(info => info.VideoType == VideoType.Mp4 && info.Resolution == 360);

/*
 * If the video has a decrypted signature, decipher it
 */
if (video.RequiresDecryption)
{
    DownloadUrlResolver.DecryptDownloadUrl(video);
}

/*
 * Create the video downloader.
 * The first argument is the video to download.
 * The second argument is the path to save the video file.
 */
var videoDownloader = new VideoDownloader(video, Path.Combine("D:/Downloads", video.Title + video.VideoExtension));

// Register the ProgressChanged event and print the current progress
videoDownloader.DownloadProgressChanged += (sender, args) => Console.WriteLine(args.ProgressPercentage);

/*
 * Execute the video downloader.
 * For GUI applications note, that this method runs synchronously.
 */
videoDownloader.Execute();
```

Extract Youtube

```
VideoInfo video = videoInfos
    .Where(info => info.CanExtractAudio)
    .OrderByDescending(info => info.AudioBitrate)
    .First();

/*
 * If the video has a decrypted signature, decipher it
 */
if (video.RequiresDecryption)
{
    DownloadUrlResolver.DecryptDownloadUrl(video);
}
```

Extract Youtube

```
private void Video_download_Click(object sender, EventArgs e)
{
    DownloadBar.Minimum = 0;
    DownloadBar.Maximum = 100;

    IEnumerable<VideoInfo> videos = DownloadUrlResolver.GetDownloadUrls(txtURL.Text);
    VideoInfo video = videos.First(p => p.VideoType == VideoType.Mp4 && p.Resolution == Convert.ToInt32(selectedIndex));
    if (video.RequiresDecryption)
        DownloadUrlResolver.DecryptDownloadUrl(video);
    VideoDownloader downloader = new VideoDownloader(video, Path.Combine(Application.StartupPath + "\\", video.Title + video.VideoExtension));
    downloader.DownloadProgressChanged += DownloadBar_DownloadProgressChanged;
    Thread thread = new Thread(() => { downloader.Execute(); }) { IsBackground = true };
    thread.Start();

}
```

```
using VideoLibrary;
```

```
using (FolderBrowserDialog fdg = new FolderBrowserDialog() { Description = "select your path" })
    if (fdg.ShowDialog() == DialogResult.OK)
    {
        var youtube = YouTube.Default;
        var video = await youtube.GetVideoAsync(txtURL.Text);
        File.WriteAllBytes(fdg.SelectedPath + video.FullName, await video.GetBytesAsync());

    }
MessageBox.Show("Completed");
```

Extract WAV

NAudio Overview

IMPORTANT The latest NAudio source code can now be found on GitHub. For now, CodePlex remains the place to access documentation, discussions and downloads.

NAudio is an open source .NET audio and MIDI library, containing dozens of useful audio related classes intended to speed development of audio related utilities in .NET. It has been in development since 2002 and has grown to include a wide variety of features. While some parts of the library are relatively new and incomplete, the more mature features have undergone extensive testing and can be quickly used to add audio capabilities to an existing .NET application. NAudio can be quickly added to your .NET application using NuGet.

Extract WAV

How do I...?

The best way to learn how to use NAudio is to download the source code and look at the two demo applications - NAudioDemo and NAudioWpfDemo. These demonstrate several of the key capabilities of the NAudio framework. They also have the advantage of being kept up to date, whilst some of the tutorials you will find on the internet refer to old versions of NAudio.

- Choose an audio output driver
- Play an MP3 File
- Understand how to convert between any audio formats you have codecs for
- Convert an MP3 to WAV
- Encode to MP3 or other formats using MediaFoundationEncoder
- Implement "Fire and Forget" Playback (e.g. game sound effects)
- Play a WAV File
- Use the WavFileWriter class
- Play streaming MP3
- Implement Looped Playback
- Trim a WAV File
- Play a Sine Wave
- Merge MP3 Files
- Convert an AIFF file to WAV
- Work with Multi-Channel Audio
- Create an ID3v2 tag
- Play and Record audio at the same time
- Fix the NoDriver calling acmFormatSuggest issue?
- Resample Audio
- Input driven Audio Resampling

Extract WAV

```
OpenFileDialog open = new OpenFileDialog();
open.Filter = "MP4 File (*.mp4)|*.mp4;";
if (open.ShowDialog() != DialogResult.OK) return;
var outputFile = @"C:\Users\송명진\Desktop\save.wav";
using (MediaFoundationReader reader = new MediaFoundationReader(open.FileName))
using (ResamplerDmoStream resampledReader = new ResamplerDmoStream(reader,
   new WaveFormat(reader.WaveFormat.SampleRate, reader.WaveFormat.BitsPerSample, reader.WaveFormat.Channels)))
// create WAVe file

using (WaveFileWriter waveWriter = new WaveFileWriter(outputFile, resampledReader.WaveFormat))
{
    // copy samples
    resampledReader.CopyTo(waveWriter);
}
```

Send Azure

```
private string itemSelected;
private string selectedIndex;
public string FileDir;
private DataRecognitionClient m_dataClient;
private SpeechRecognitionMode m_recoMode = SpeechRecognitionMode.LongDictation;
public string[] DefaultLocale = { "en-US", "ja-JP", "ko-KR", "zh-CN" };
public string[] resolution = { "240", "360", "480" };
private string primaryKey2 = "4154a6488c024226a13ecab7f3b3207f";
private const string Translate_SubscriptionKey = "61e621efd253434f8c5462e4de4f1249";
```

```
private void initalize()
{
    m_dataClient = SpeechRecognitionServiceFactory.CreateDataClient(m_recoMode, itemSelected, primaryKey2);

    // Event handlers for speech recognition results
    m_dataClient.OnResponseReceived += this.OnResponseReceivedHandler;
    m_dataClient.OnPartialResponseReceived += this.OnPartialResponseReceivedHandler;
    m_dataClient.OnConversationError += this.OnConversationErrorHandler;

    Language.Items.AddRange(DefaultLocale);
    Language.SelectedIndex = 0;

    selectResolution.Items.AddRange(resolution);
    selectResolution.SelectedIndex = 0;
}
```

Send Azure

```
private void OnResponseReceivedHandler(object sender, SpeechResponseEventArgs e)
{
    bool isFinalDicationMessage = m_recoMode == SpeechRecognitionMode.LongDictation &&
                                  (e.PhraseResponse.RecognitionStatus == RecognitionStatus.EndOfDictation ||
                                   e.PhraseResponse.RecognitionStatus == RecognitionStatus.DictationEndSilenceTimeout);

    if (!isFinalDicationMessage)
    {
        this.WriteLine("********* Final NBEST Results *********");

        for (int i = 0; i < e.PhraseResponse.Results.Length; i++)
        {
            this.WriteLine("[{0}] Confidence={1} Text=\"{2}\"",
                           i, e.PhraseResponse.Results[i].Confidence,
                           e.PhraseResponse.Results[i].DisplayText);


            using (System.IO.StreamWriter file = new System.IO.StreamWriter(FileDir, true))
            {
                file.WriteLine(e.PhraseResponse.Results[i].DisplayText);
            }
        }
        this.WriteLine();
    }
}
```

Send Azure

```
private void SendAudioHelper(string wavFileName)
{
    using (FileStream fileStream = new FileStream(wavFileName, FileMode.Open, FileAccess.Read))
    {
        int bytesRead = 0;
        byte[] buffer = new byte[1024];

        try
        {
            do
            {
                // Get more Audio data to send into byte buffer.
                bytesRead = fileStream.Read(buffer, 0, buffer.Length);

                // Send of audio data to service.
                this.m_dataClient.SendAudio(buffer, bytesRead);
            }
            while (bytesRead > 0);
        }
        finally
        {
            this.m_dataClient.EndAudio();
        }
    }
}
```

Send Azure

Request

Method	Request URI
GET	`http://api.microsofttranslator.com/V2/Http.svc/Translate`

Parameters

Parameter	Description
appId	**Required.** If the Authorization header is used, leave the appid field empty else a string containing "Bearer" + " " + access token.
text	**Required.** A string representing the text to translate. The size of the text must not exceed 10000 characters.
from	**Optional.** A string representing the language code of the translation text.
to	**Required.** A string representing the language code to translate the text into.
contentType	**Optional.** The format of the text being translated. The supported formats are "text/plain" and "text/html". Any HTML needs to be well-formed.
category	**Optional.** A string containing the category (domain) of the translation. Defaults to "general".

Send Azure

```
GET wss://dev.microsofttranslator.com/speech/translate?from=en-US&to=it-IT&features=texttospeech&voice=it-IT-Elsa&api-version=1.0
Authorization: Bearer {access token}
X-ClientTraceId: {GUID}
```

```
private void TRANSLATOR_Click(object sender, EventArgs e)
{
    AdmAccessToken admToken;
    string headerValue;
    AdmAuthentication admAuth = new AdmAuthentication("reki318", "0LZCC/QenWjOVb7vuBdXJ5yX5jI/69uQGnAVzF2VYvw=");

    try
    {
        admToken = admAuth.GetAccessToken();
        headerValue = "Bearer " + admToken.access_token;
        TransformTextMethod(headerValue);
    }
    catch (WebException e1)
    {
        ProcessWebException(e1);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

Send Azure

```
private static void TransformTextMethod(string authToken)
    {
        string text = System.IO.File.ReadAllText(@"C:\Users\송명진\Desktop\대본.txt");
        string from = "en";
        string to = "ko";

        string uri = "http://api.microsofttranslator.com/v2/Http.svc/Translate?text=" + System.Web.HttpUtility.UrlEncode(text) + "&from=" + from +

        HttpWebRequest httpWebRequest = (HttpWebRequest)WebRequest.Create(uri);
        httpWebRequest.Headers.Add("Authorization", authToken);

        WebResponse response = null;
        try
        {
            response = httpWebRequest.GetResponse();
            using (Stream stream = response.GetResponseStream())
            {
                System.Runtime.Serialization.DataContractSerializer dcs = new System.Runtime.Serialization.DataContractSerializer(Type.GetType("Sy
                string translation = (string)dcs.ReadObject(stream);
                Console.WriteLine("Translation for source text '{0}' from {1} to {2} is", text, "en", "de");
                using (System.IO.StreamWriter file = new System.IO.StreamWriter(@"C:\Users\송명진\Desktop\대본_번역.txt", true))
                {
                    file.WriteLine(translation);
                }
            }
        }
        catch(HttpRequestException e)
        {
            Console.WriteLine(e);
        }
```

Microsoft