Translate Lab

Clone the repository for this course

1. Start Visual Studio Code
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/FmoOliveira/ITArchWeek2022` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.
4. Wait while additional files are installed to support the C# code projects in the repo.

   Note: If you are prompted to add required assets to build and debug, select **Not Now**.

Prepare to use the Speech Translation service

In this exercise, you'll complete a partially implemented client application that uses the Speech SDK to recognize, translate, and synthesize speech.

1. In Visual Studio Code, in the **Explorer** pane, browse to the **Translator** folder and expand it.
2. Right-click the **Translator** folder and open an integrated terminal. Then install the *Speech SDK* package by running the appropriate command:

   ```
   dotnet add package Microsoft.CognitiveServices.Speech --version 1.19.0
   ```

3. View the contents of the **Translator** folder, and note that it contains a file for configuration settings: *appsettings.json*. Open the configuration file and update the configuration values it contains to include an authentication **key** for your cognitive services resource, and the **location** where it is deployed. Save your changes.

   1. KEY: ##########
   2. Location: westeurope
4. Note that the **Translator** folder contains a code file for the client application: *Program.cs*

   1. Open the code file and at the top, under the existing namespace references, find the comment **Import namespaces**. Then, under this comment, add the following language-specific code to import the namespaces you will need to use the Speech SDK:
   2. ```
      // Import namespaces
      using Microsoft.CognitiveServices.Speech;
      using Microsoft.CognitiveServices.Speech.Audio;
      using Microsoft.CognitiveServices.Speech.Translation;
      ```

5. In the **Main** function, note that code to load the cognitive services key and region from the configuration file has already been provided. You must use these variables to create a **SpeechTranslationConfig** for your cognitive services resource, which you will use to translate spoken input. Add the following code under the comment **Configure translation**:
      ```

```
 // Configure translation
 translationConfig = SpeechTranslationConfig.FromSubscription(cogSvcKey,
cogSvcRegion);
 translationConfig.SpeechRecognitionLanguage = "en-US";
 translationConfig.AddTargetLanguage("fr");
 translationConfig.AddTargetLanguage("es");
 translationConfig.AddTargetLanguage("de");
 Console.WriteLine("Ready to translate from " +
translationConfig.SpeechRecognitionLanguage);
```

6. You will use the **SpeechTranslationConfig** to translate speech into text, but you will also use a **SpeechConfig** to synthesize translations into speech. Add the following code under the comment **Configure speech**:

```
 // Configure speech
 speechConfig = SpeechConfig.FromSubscription(cogSvcKey, cogSvcRegion);
```

7. Save your changes and return to the integrated terminal for the **translator** folder, and enter the following command to run the program:

```
 dotnet run
```

8. Ignore any warnings about using the **await** operator in asynchronous methods - we'll fix that later. The code should display a message that it is ready to translate from en-US. Press ENTER to end the program.

Implement speech translation

Now that you have a **SpeechTranslationConfig** for the speech service in your cognitive services resource, you can use the **Speech translation** API to recognize and translate speech.

If you have a working microphone

1. In the **Main** function for your program, note that the code uses the **Translate** function to translate spoken input.
2. In the **Translate** function, under the comment **Translate speech**, add the following code to create a **TranslationRecognizer** client that can be used to recognize and translate speech using the default system microphone for input.

```
 // Translate speech
 using AudioConfig audioConfig = AudioConfig.FromDefaultMicrophoneInput();
 using TranslationRecognizer translator = new
TranslationRecognizer(translationConfig, audioConfig);
 Console.WriteLine("Speak now...");
 TranslationRecognitionResult result = await
translator.RecognizeOnceAsync();
 Console.WriteLine($"Translating '{result.Text}'");
 translation = result.Translations[targetLanguage];
 Console.OutputEncoding = Encoding.UTF8;
 Console.WriteLine(translation);
```

> **Note**: The code in your application translates the input to all three languages in a single call. Only the translation for the specific language is displayed, but you could retrieve any of the translations by specifying the target language code in the translations collection of the result.

3. Now skip ahead to the **Run the program** section below.

Alternatively, use audio input from a file

1. In the terminal window, enter the following command to install a library that you can use to play the audio file:

```
dotnet add package System.Windows.Extensions --version 4.6.0
```

2. In the code file for your program, under the existing namespace imports, add the following code to import the library you just installed:

```
using System.Media;
```

3. In the **Main** function for your program, note that the code uses the **Translate** function to translate spoken input. Then in the **Translate** function, under the comment **Translate speech**, add the following code to create a **TranslationRecognizer** client that can be used to recognize and translate speech from a file.

```
// Translate speech
string audioFile = "station.wav";
SoundPlayer wavPlayer = new SoundPlayer(audioFile);
wavPlayer.Play();
using AudioConfig audioConfig = AudioConfig.FromWavFileInput(audioFile);
using TranslationRecognizer translator = new
TranslationRecognizer(translationConfig, audioConfig);
Console.WriteLine("Getting speech from file...");
TranslationRecognitionResult result = await
translator.RecognizeOnceAsync();
Console.WriteLine($"Translating '{result.Text}'");
translation = result.Translations[targetLanguage];
Console.OutputEncoding = Encoding.UTF8;
Console.WriteLine(translation);
```

Run the program

1. Save your changes and return to the integrated terminal for the **translator** folder, and enter the following command to run the program:

```
dotnet run
```

2. When prompted, enter a valid language code (*fr*, *es*, or *hi*), and then, if using a microphone, speak clearly and say “where is the station?” or some other phrase you might use when traveling abroad. The program should transcribe your spoken input and translate it to the language you specified (French, Spanish, or German). Repeat this process, trying each

language supported by the application. When you're finished, press ENTER to end the program.

> **Note**: The TranslationRecognizer gives you around 5 seconds to speak. If it detects no spoken input, it produces a "No match" result.
>
> The translation to Hindi may not always be displayed correctly in the Console window due to character encoding issues.

Synthesize the translation to speech

So far, your application translates spoken input to text; which might be sufficient if you need to ask someone for help while traveling. However, it would be better to have the translation spoken aloud in a suitable voice.

1. In the **Translate** function, under the comment **Synthesize translation**, add the following code to use a **SpeechSynthesizer** client to synthesize the translation as speech through the default speaker:

```
// Synthesize translation
var voices = new Dictionary<string, string>
                {
                    ["fr"] = "fr-FR-HenriNeural",
                    ["es"] = "es-ES-ElviraNeural",
                    ["de"] = "de-DE-KatjaNeural"
                };
speechConfig.SpeechSynthesisVoiceName = voices[targetLanguage];
using SpeechSynthesizer speechSynthesizer = new
SpeechSynthesizer(speechConfig);
SpeechSynthesisResult speak = await
speechSynthesizer.SpeakTextAsync(translation);
if (speak.Reason != ResultReason.SynthesizingAudioCompleted)
{
    Console.WriteLine(speak.Reason);
}
```

2. Save your changes and return to the integrated terminal for the **translator** folder, and enter the following command to run the program:

```
dotnet run
```

3. When prompted, enter a valid language code (*fr*, *es*, or *hi*), and then speak clearly into the microphone and say a phrase you might use when traveling abroad. The program should transcribe your spoken input and respond with a spoken translation. Repeat this process, trying each language supported by the application. When you're finished, press ENTER to end the program.

> **Note** *In this example, you've used a **SpeechTranslationConfig** to translate speech to text, and then used a **SpeechConfig** to synthesize the translation as speech. You can in fact use the **SpeechTranslationConfig** to synthesize the translation directly, but this only works when translating to a single language, and results in an audio stream that is typically saved as a file rather than sent directly to a speaker.*

More information

For more information about using the **Speech translation** API, see the Speech translation documentation.