Speech 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 service

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

1. In Visual Studio Code, in the **Explorer** pane, browse to the **Speech** folder and expand it.
2. Right-click the **Speech** 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 **Speech** 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 **Speech** 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;
      ```

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 **SpeechConfig** for your cognitive services resource. Add the following code under the comment **Configure speech service**:
      ```

```
// Configure speech service
speechConfig = SpeechConfig.FromSubscription(cogSvcKey, cogSvcRegion);
Console.WriteLine("Ready to use speech service in " + speechConfig.Region);

// Configure voice
speechConfig.SpeechSynthesisVoiceName = "en-US-AriaNeural";
```

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

```
dotnet run
```

7. Ignore any warnings about using the **await** operator in asynchronous methods - we'll fix that later. The code should display the region of the speech service resource the application will use

Recognize speech

Now that you have a **SpeechConfig** for the speech service in your cognitive services resource, you can use the **Speech-to-text** API to recognize speech and transcribe it to text.

If you have a working microphone

In the **Main** function for your program, note that the code uses the **TranscribeCommand** function to accept spoken input.

1. In the **TranscribeCommand** function, under the comment **Configure speech recognition**, add the appropriate code below to create a **SpeechRecognizer** client that can be used to recognize and transcribe speech using the default system microphone:

```
// Configure speech recognition
using AudioConfig audioConfig = AudioConfig.FromDefaultMicrophoneInput();
using SpeechRecognizer speechRecognizer = new
SpeechRecognizer(speechConfig, audioConfig);
Console.WriteLine("Speak now...");
```

Now skip ahead to the **Add code to process the transcribed command** 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, note that the code uses the TranscribeCommand function to accept spoken input. Then in the TranscribeCommand function, under the comment Configure speech recognition, add the appropriate code below to create a SpeechRecognizer client that can be used to recognize and transcribe speech from an audio file:

```
 // Configure speech recognition
string audioFile = "time.wav";
SoundPlayer wavPlayer = new SoundPlayer(audioFile);
wavPlayer.Play();
using AudioConfig audioConfig = AudioConfig.FromWavFileInput(audioFile);
using SpeechRecognizer speechRecognizer = new
SpeechRecognizer(speechConfig, audioConfig);
```

Add code to process the transcribed command

1. In the **TranscribeCommand** function, under the comment **Process speech input**, add the following code to listen for spoken input, being careful not to replace the code at the end of the function that returns the command:

```
// Process speech input
SpeechRecognitionResult speech = await
speechRecognizer.RecognizeOnceAsync();
if (speech.Reason == ResultReason.RecognizedSpeech)
{
    command = speech.Text;
    Console.WriteLine(command);
}
else
{
    Console.WriteLine(speech.Reason);
    if (speech.Reason == ResultReason.Canceled)
    {
        var cancellation = CancellationDetails.FromResult(speech);
        Console.WriteLine(cancellation.Reason);
        Console.WriteLine(cancellation.ErrorDetails);
    }
}
```

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

```
dotnet run
```

3. If using a microphone, speak clearly and say “what time is it?”. The program should transcribe your spoken input and display the time (based on the local time of the computer where the code is running, which may not be the correct time where you are).

   The SpeechRecognizer gives you around 5 seconds to speak. If it detects no spoken input, it produces a “No match” result.

If the SpeechRecognizer encounters an error, it produces a result of "Cancelled". The code in the application will then display the error message. The most likely cause is an incorrect key or region in the configuration file.

Synthesize speech

Your application accepts spoken input, but it doesn't actually speak! Let's fix that by adding code to synthesize speech.

1. In the **Main** function for your program, note that the code uses the **TellTime** function to tell the user the current time.
2. In the **TellTime** function, under the comment **Configure speech synthesis**, add the following code to create a **SpeechSynthesizer** client that can be used to generate spoken output:

```
// Configure speech synthesis
speechConfig.SpeechSynthesisVoiceName = "en-GB-RyanNeural";
using SpeechSynthesizer speechSynthesizer = new
SpeechSynthesizer(speechConfig);
```

3. In the **TellTime** function, under the comment **Synthesize spoken output**, add the following code to generate spoken output, being careful not to replace the code at the end of the function that prints the response:

```
// Synthesize spoken output
SpeechSynthesisResult speak = await
speechSynthesizer.SpeakTextAsync(responseText);
if (speak.Reason != ResultReason.SynthesizingAudioCompleted)
{
    Console.WriteLine(speak.Reason);
}
```

4. Save your changes and return to the integrated terminal for the **speaking-clock** folder, and enter the following command to run the program:

```
dotnet run
```

5. When prompted, speak clearly into the microphone and say "what time is it?". The program should speak, telling you the time.

Use a different voice

Your application uses a default voice, which you can change. The Speech service supports a range of *standard* voices as well as more human-like *neural* voices. You can also create *custom* voices.

Note: For a list of neural and standard voices, see Language and voice support List in the Speech service documentation.

1. In the **TellTime** function, under the comment **Configure speech synthesis**, modify the code as follows to specify an alternative voice before creating the **SpeechSynthesizer** client:

```
 // Configure speech synthesis
 speechConfig.SpeechSynthesisVoiceName = "en-GB-LibbyNeural"; // change this
 using SpeechSynthesizer speechSynthesizer = new
SpeechSynthesizer(speechConfig);
```

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

   ```
   dotnet run
   ```

3. When prompted, speak clearly into the microphone and say "what time is it?". The program should speak in the specified voice, telling you the time.

More information

For more information about using the **Speech-to-text** and **Text-to-speech** APIs, see the Speech-to-text documentation and Text-to-speech documentation.