# Créer un service natif de récupération de synthèse vocale

# Démonstration 1 du module 7

|  |
| --- |
| Manipuler les différents types de layouts (StackLayout, RelativeLayout et GridLayout) et comprendre les propriétés d’alignement et de répartition de l’espace vide |

Déroulement

* Créer une interface dans le projet .NET Standard **ITextToSpeech** en effectuant un clic droit sur le projet et en sélectionnant **Ajouter** - **Nouvel élément** - **Interface Visual C#**

public interface ITextToSpeed

{

void Speak(string text);

}

* Ajouter une méthode publique Speak qui prend en paramètre une chaîne de caractères

public interface ITextToSpeed

{

void Speak(string text);

}

* Ajouter une classe C# **TextToSpeechImplementation** au projet iOS qui implémente l’interface **ITextToSpeech**

[assembly: Dependency(typeof(TextToSpeechImplementation))]

namespace DependencyServiceSample.iOS

{

public class TextToSpeechImplementation : ITextToSpeech

{

public TextToSpeechImplementation() { }

public void Speak(string text)

{

var speechSynthesizer = new AVSpeechSynthesizer();

var speechUtterance = new AVSpeechUtterance(text)

{

Rate = AVSpeechUtterance.MaximumSpeechRate / 4,

Voice = AVSpeechSynthesisVoice.FromLanguage("en-US"),

Volume = 0.5f,

PitchMultiplier = 1.0f

};

speechSynthesizer.SpeakUtterance(speechUtterance);

}

}

}

* Ajouter une classe C# **TextToSpeechImplementation** au projet Android qui implémente l’interface **ITextToSpeech**

[assembly: Dependency(typeof(TextToSpeechImplementation))]

namespace TPXamarin.Droid

{

public class TextToSpeechImplementation : Java.Lang.Object, ITextToSpeech, TextToSpeech.IOnInitListener

{

TextToSpeech speaker;

string toSpeak;

public void Speak(string text)

{

toSpeak = text;

if (speaker == null)

{

speaker = new TextToSpeech(Forms.Context as Activity, this);

}

else

{

speaker.Speak(toSpeak, QueueMode.Flush, null, null);

}

}

public void OnInit(OperationResult status)

{

if (status.Equals(OperationResult.Success))

{

speaker.Speak(toSpeak, QueueMode.Flush, null, null);

}

}

}

}

* Ajouter une classe C# **TextToSpeechImplementation** au projet Windows qui implémente l’interface **ITextToSpeech**

[assembly: Dependency(typeof(TextToSpeechImplementation))]

namespace TPXamarin.UWP

{

public class TextToSpeechImplementation : ITextToSpeech

{

public async void Speak(string text)

{

var mediaElement = new MediaElement();

var synth = new Windows.Media.SpeechSynthesis.SpeechSynthesizer();

var stream = await synth.SynthesizeTextToStreamAsync(text);

mediaElement.SetSource(stream, stream.ContentType);

mediaElement.Play();

}

}

}

* Ajouter un Entry et un bouton dans la page MainPage.xaml qui, lorsqu’il est cliqué, appelle une méthode **Parler\_Clicked**

<StackLayout Orientation="Vertical">

<Entry Placeholder="Texte à dicter" HorizontalOptions="FillAndExpand"></Entry>

<Button Text="Parler" HorizontalOptions="FillAndExpand" Clicked="Parler\_Clicked"></Button>

</StackLayout>

* Appeler le service natif TextToSpeech au clic sur le bouton

private void Parler\_Clicked(object sender, EventArgs e)

{

DependencyService.Get<ITextToSpeech>().Speak(this.texteADicter.Text.ToString());

}