Usando SSML para text-to-speech avanzado en Windows Phone 8

Este artículo demuestra como el lenguaje SSML (del Inglés Speech Synthesis Markup Language) puede ser usado para proveer funcionalidad avanzada de *text-to-speech* (TTS) en aplicaciones de Windows Phone 8.

Introducción

Microsoft introdujo un número de APIs y características interesantes en el campo de voz en Windows Phone 8, las cuales incluyen comandos de voz, reconocimiento de voz y *text-to-speech* (TTS).

Aunque este artículo explica cómo agregar funcionalidad TTS básica a una aplicación, principalmente se enfoca en el uso del lenguaje SSML que es soportado por Windows Phone 8, para permitir un uso más avanzado del motor de *text-to-speech*.

Note: En el artículo se utilizará texto en español para algunos de los ejemplos, pero en la aplicación descargable que lo acompaña, todos los textos se encuentran en Inglés

Prerequisitos

Este artículo asume un conocimiento básico del IDE Visual Studio, lo cual incluye conocimientos sobre cómo crear, compilar y correr aplicaciones, cómo agregar subdirectorios y crear y/o agregar elementos nuevos o previamente existentes a ellas, y un conocimiento básico del lenguaje C# y programación orientada a objetos en general.

TTS en Windows Phone 8

Warning: Para usar TTS, se debe setear la capacidad ID_CAP_SPEECH_RECOGNITION en el manifiesto de la aplicación. Si no se configura esta capacidad, la aplicación puede no funcionar correctamente, o generar errores en tiempo de corrida.

Para agregar funcionalidad TTS a una aplicación de Windows Phone 8, lo primero que se necesita hacer es incluir una sentencia `using` para el espacio de nombres de síntesis de voz. Esto es necesario en cualquier aplicación con funcionalidad TTS:

```
using Windows.Phone.Speech.Synthesis;
```

Luego, un nuevo objeto de la clase SpeechSynthesizer debe ser instanciado:

```
SpeechSynthesizer synth = new SpeechSynthesizer();
```

Y si solamente una funcionalidad básica de TTS es necesaria, sólo se necesita un paso adicional, el cual es una llamada al método `SpeakTextAsync` para 'hablar' el texto:

```
await synth.SpeakTextAsync("Probando el TTS en WP8");
```

Tip: El operador `await` es usado en conjunto con métodos asíncronos para asegurarse de que las aplicaciones permanezcan responsivas. (Ver Asynchronous Programming with Async and Await y Asynchronous Programming For Windows Phone 8 (ambos artículos en Inglés))

Este uso básico del motor de *text-to-speech* puede ser suficiente para algunas aplicaciones, pero Windows Phone 8 provee mecanismos y métodos adicionales para agregar una funcionalidad de TTS más avanzada, por medio de la utilización del lenguaje SSML.

Lenguaje SSML (Speech Synthesis Markup Language)

SSML es un lenguaje basado en XML para aplicaciones de síntesis de voz, y es una recomendación del W3C's voice browser working group. Permite a los desarrolladores de aplicaciones controlar varias características de la voz sintetizada, como por ejemplo la voz, el lenguaje, la pronunciación, etc.

La implementación de SSML de Microsoft está basada en la definición Speech Synthesis Markup Language (SSML) Version 1.0. del World Wide Web Consortium.

La clase `SpeechSynthesizer` provee 2 métodos para 'hablar' texto que incluya lenguaje SSML. El primero, `SpeakSsmlAsync`, recibe el texto a hablar como un parámetro, y el segundo, `SpeakSsmlFromUriAsync`, 'habla' el texto de un archivo externo de SSML. El primero de estos será utilizado para mostrar las diferentes características de voz que pueden ser controladas por medio de SSML, y al final del artículo se incluirá una corta explicación sobre como 'hablar' el contenido de un documento SSML externo.

Cada documento o string de SSML require un elemento speak. Este es el elemento raiz del documento, y puede ser usado sin ningún otro elemento. El

(C) Copyright Nokia 2013. All rights reserved.

elemento speak también especifica el lenguaje que sera utilizado.

Esta es la sintaxis:

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="string"> </speak>
```

Los tres atributos (version, xmlns y xml:lang) son obligatorios.

Aquí hay un ejemplo:

```
synth = new SpeechSynthesizer();
string ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"es\">";
ssmlText += " Probando el TTS de Windows Phone 8";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

Usar SSML en ésta forma produce los mismos resultados que la forma básica de hacer aplicaciones 'hablar' descrita al principio del artículo, pero hay muchos otros elementos que pueden ser usados para agregar características más avanzadas, como las que se esplican a continuación.

Insertando archivos de sonido

El elemento audio puede ser usado para inserter archivos de sonido en el discurso. Este soporta archivos en formato PCM, a-law y u-law format, con 8 o 16 bits de profundidad, y no-estéreo (solamente mono).

Esta es la sintaxis:

```
<audio src="string" /></audio>
```

Y aquí está un ejemplo:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"es\">";
ssmlText += "Aquí viene el perro, <audio src=\"ms-appx:///Assets/cats.wav\">Perro </audio>";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

En este ejemplo, el archivo **cats.wav** que está ubicado en el subdirectorio **Assets** de la aplicación, sera ejecutado después de decir el texto "Aquí viene el perro". Si el archivo no cumple con los formatos soportados, o si por cualquier razón el archivo no puede ser ejecutado, el texto que está incluído entre las etiquetas <audio> y </audio> sera el que se 'hable' (en este caso "Perro").

> **Tip:** La ubicación del archivo de sonido se podría haber definido simplemente como "**Assets/cats.wav**" en lugar de "**ms-appx:///Assets/cats.wav**". La diferencia radica en la inclusión del URI scheme. Esta es la forma recomendada de hacerlo, ya que en algunas circunstancias la aplicación podría fallar al tratar de localizar los archivos si el esquema (scheme) no es incluído. El esquema ms-appx es usado para referirse a archivos que vienen del paquete de la aplicación.

Insertando pausas

El elemento break puede usarse para inserter pausas.

Esta es la sintaxis:

```
<break />
<break strength="string" />
<break time="string" />
```

- El atributo strength es opcional, y puede aceptar cualquiera de estos valores: none, x-weak, weak, medium, strong, o x-strong.
- El atributo time también es opcional, y es usado para definer la duración de la pausa en segundos o milisegundos.

Aquí hay un ejemplo:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"es\">";
ssmlText += "Hay una pausa<break time=\"500ms\" /> aquí, y otra <break strength=\"x-strong\" /> aquí";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

Definiendo o cambiando la pronunciación de las palabras

Hay dos formas de instruir al sintetizador de voz para que pronuncie una palabra de una manera específica. Para definir una pronuncación de una sóla vez para una palabra, el elemento phoneme es una buena opción, pero para palabras que deben ser pronunciadas de una forma específica cada vez que aparecen en el mismo documento, o para definir la pronunciación de más de una palabra en un mismo lugar, el elemento lexicon es la opción correcta.

Es importante tomar en cuenta que cuando un documento incluye los elementos phoneme y lexicon, el sintetizador de voz le da precedencia a los elementos phoneme.

Los fonemas pueden ser definidos en-linea, como los otros elementos que se han mencionado en este artículo.

Esta es la sintaxis:

(C) Copyright Nokia 2013. All rights reserved.

```
<phoneme alphabet="string" ph="string">palabra</phoneme>
```

La pronunciación de la palabra "word" se define en el atributo `ph`, de acuerdo al `alphabet` especificado.

Aquí está un ejemplo:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"en-US\">";
ssmlText += "<phoneme alphabet=\"x-microsoft-ups\" ph=\"O L AA\">hello</phoneme>, I mean hello";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

El uso del elemento `lexicon` require un paso adicional, el cual es la creación y/o adición de un archivo externo de lexicon.

Los archivos de lexicon son documentos XML que contienen el mapeo entre las pronunciaciones y la representación escrita de las palabras, de acuerdo a una especificación predefinida.

Para este ejemplo el archivo de lexicon usado se llama **lexicon1.xml**, e incluye el siguiente texto (en Inglés):

```
<?xml version="1.0" encoding="UTF-8"?>
<lexicon version="1.0"
  xmlns="http://www.w3.org/2005/01/pronunciation-lexicon"
  alphabet="x-microsoft-ups" xml:lang="en-US">
  <lexeme>
    <grapheme>wife</grapheme>
    <phoneme> W AI F AI</phoneme>
  </lexeme>
</lexicon>
```

Cada palabra se define en un elemento `lexeme`, cuyo contenido incluye un `grapheme` (la palabra para la cual la pronuncicación está siendo definida) y un `phoneme` (la definición de cómo debe ser pronunciada la palabra de acuerdo al `alphabet` especificado en su elemento `lexicon` raíz).

En este ejemplo, estamos instruyendo al sintetizador de voz a que pronuncie la palabra "wife" como "WiFi".

Aquí está un ejemplo de cómo se puede utilizar este lexicon:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"en-US\">";
ssmlText += "<lexicon uri=\"ms-appx:///Assets/lexicon1.xml\" type=\"application/pls+xml\"/>";
ssmlText += "She is not my wife";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

Es importante tomar en cuenta que el lexicon no sera usado si lenguaje seleccionado actualmente es diferente al especificado en el lexicon, por ejemplo, si un lenguaje diferente fue especificado por medio de código usando el método `SetVoice` de la clase `SpeechSynthesizer`, o si un lenguaje diferente fue definido en el elemento `speak` raiz del documento SSML.

Note: La documentación oficial de Microsoft sobre el elemento `lexicon` (la cual no es específica para Windows Phone, (link)) usada para investigar y probar algunos de los temas incluídos en este artículo, incluye un ejemplo de un archivo lexicon con extensión **pls**, pero usar esa extensión causa una excepción `IO.FileNotFoundException`, mientras que usar archivos con extensión XML funciona bien.

Cambiando la voz usada para hablar

Hay varias formas de cambiar la voz que el sintetizador de voz usa para 'hablar' el texto.

La voz puede ser cambiada por medio de código, usando el método `SetVoice` de la clase `SpeechSynthesizer`.

Aquí hay un ejemplo que usa un query LINQ para buscar una voz femenina que hable Inglés británico.

```
IEnumerable<VoiceInformation> englishVoices = from voice in InstalledVoices.All
                                    where voice.Language == "en-GB"
                                    && voice.Gender.Equals(VoiceGender.Female)
if(englishVoices.Count() > 0)
    synth.SetVoice(englishVoices.ElementAt(0));
await synth.SpeakTextAsync("Testing Windows Phone 8 TTS");
```

La voz también puede ser cambiada usando el elemento SSML voice.

Esta es la sintaxis:

```
<voice name="string" gender="string" age="integer" xml:lang="string" variant="integer"> </voice>
```

Todos los atributos son opcionales, pero al menos uno debe ser incluído. Es importante mencionar que la mayoría de estos atributos son considerados "valores preferidos o deseados" por el sintetizador de voz, así que si una voz apropiada que cumpla con esos valores no está disponible, una voz con diferentes atributos será usada.

Aquí hay un ejemplo:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"en-US\">";
ssmlText += "<voice name=\"Microsoft Susan Mobile\" gender=\"female\" age=\"30\" xml:lang=\"en-US\">";
```


(C) Copyright Nokia 2013. All rights reserved.

```
ssmlText += "This is another test </voice>";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

Adicionalmente, el lenguaje de la voz puede cambiarse usando los elementos p and s, los cuales son usados para marcar la estructura de oraciones y párrafos en el documento SSML, cuando la estructura detectada automáticamente no es satisfactoria para el desarrollador.

Esta es la sintaxis:

```
<p xml:lang="string"> </p>
<s xml:lang="string"> </s>
```

Y aquí hay un ejemplo:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"es\">";
ssmlText += "<p>";
ssmlText += "<s>Primera oración de un párrafo</s>";
ssmlText += "<s xml:lang=\"en-US\">Y esta es la segunda oración</s>";
ssmlText += "</p>";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

Observe que cada elemento que puede definir un lenguaje, puede definir uno diferente del que se definió en el elemento `speak` raiz, porque SSML soporta el uso de diferentes lenguajes en un mismo documento.

Cambiando la prosodia del discurso

Algunos cambios en la prosodia del discurso pueden hacerse agregando pausas con el elemento `break`, y definiendo la estructura del documento con los elementos `p` y `s` (los 3 han sido explicados en seciones previas), pero el elemento que provee el más fino control sobre la prosodia del discurso es el elemento prosody. Este puede ser usado para controlar el tono y el volumen de la voz, entre otras cosas.

Esta es la sintaxis:

```
<prosody pitch="value" contour="value" range="value" rate="value" duration="value" volume="value"> </prosody>
```

- El atributo `pitch` puede ser definido como un valor absoluto expresado en Hertz (por ejemplo "500Hz") o como un valor relativo, expresado como el cambio de tono en Hertz o semitonos (por ejemplo "+10Hz" o "-1st"). También puede ser dcefinido como un valor específico de una enumeración predefinida que incluye los siguientes valores: `x-low`, `low`, `medium`, `high`, `x-high`, o `default`.
- Los atributos `contour` y `range` se supone que pueden ser usados para definir diferentes tonos en diferentes partes del discurso, y para definir un rango de tonos, respectivamente, pero al momento de escribir este artículo, estos no producen ningún efecto en el discurso producido por el sintetizador de voz en Windows Phone 8.
- El atributo `rate` define la velocidad en la cual el discurso debe ser ejecutado, expresado como un múltiplo del valor default (por ejemplo "2" para usar el doble de velocidad) o como un valor específico de una enumeración que incluye estos valores: `x-slow`, `slow`, `medium`, `fast`, `x-fast`, o `default` .
- El atributo `duration` se supone que puede ser usado para alterar la velocidad del discurso también, pero al momento de escribir este artículo, este no produce ningún efecto en el discurso producido por el sintetizador de voz en Windows Phone 8.
- El atributo `volume` define el volumen, y puede tomar valores absolutos en el rango de 0.0 a 100.0, valores relativos (expresados como un cambio de volumen, por ejemplo "+5") o valores específicos de una enumeración que incluye los siguientes valores: `x-soft`, `soft`, `medium`, `loud`, `x-loud`, o `default`.

Aquí hay un ejemplo mostrando algunos de estos atributos:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"es\">";
ssmlText += "Probando el ";
ssmlText += "<prosody pitch=\"+100Hz\" volume=\"70.0\" >elemento</prosody>";
ssmlText += " prosody";
ssmlText += "Normal,<prosody rate=\"2\"> Muy rápido,</prosody>";
ssmlText += "<prosody rate=\"0.4\"> ahora despacio,</prosody>";
ssmlText += "y normal de nuevo";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

Todos los atributos del elemento `prosody` son considerados como "sugerencias" por el sintetizador de voz, y debido a eso, algunos de ellos pueden ser ignorados o substituidos si el procesador de voz no los soporta.

Monitoreando el progreso del discurso

Si una aplicación necesita tomar acciones específicas dependiendo del progreso de un discurso, ésta puede usar el elemento mark. Este es un elemento vacío que se usa para marcar una posición específica en un discurso. Cuando el sintetizador de voz está ejecutando el discurso y encuentra una marca (elemento `mark`), este dispara un evento `SpeechBookmarkReached` que incluye información para identificar el elemento `mark` que lo generó, el cual el desarrollador puede usar para ejecutar las acciones requeridas.

Esta es la sintaxis:

```
<mark name="string" />
```

Y este es un ejemplo de cómo puede ser usado:

(C) Copyright Nokia 2013. All rights reserved.

```
public MainPage()
{
    InitializeComponent();
    synth = new SpeechSynthesizer();
    // Agregar el manejador de evento para los eventos de progreso del discurso
    synth.BookmarkReached += new TypedEventHandler<SpeechSynthesizer, SpeechBookmarkReachedEventArgs>(synth_BookmarkReached);
}

private async void Button7_Click(object sender, RoutedEventArgs e)
{
    ssmlText = "<speak version=\"1.0\" ";
    ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"es\">";
    ssmlText += "<mark name=\"START\"/>";
    ssmlText += "Esa es la primera mitad del discurso.";
    ssmlText += "<mark name=\"HALF\"/>";
    ssmlText += "Y esta es la segunda mitad. Finalizando en este momento";
    ssmlText += "<mark name=\"END\"/>";
    ssmlText += "</speak>";
    await synth.SpeakSsmlAsync(ssmlText);
}

static void synth_BookmarkReached(object sender, SpeechBookmarkReachedEventArgs e)
{
    Debugger.Log(1, "Info", "Marca " + e.Bookmark + " encontrada\n");
}
```

- Los espacios de nombres `Windows.Foundation` y `System.Diagnostics` necesitan ser usados para que este código compile.

Para apreciar los resultados de este ejemplo, presione F5 para debugear su aplicación y ponga atención a la ventana de salida (output) mientras hace click en el botón para iniciar el discurso.

Observe que dos pasos adicionales fueron necesarios para aprovechar esta capacidad de monitoreo del progreso: Primero, un nuevo manejador de eventos con tipo (typed events) fue agregado al evento `BookmarkReached` del sintetizador, y luego el método manejador del evento en sí, tuvo que ser implementado.

Especificando el tipo de contenido y asignando sobrenombres o aliases a ciertas partes del discurso

El elemento say-as puede ser usado para especificar el tipo de contenido del texto incluido en el elemento, de forma que el sintetizador sepa cómo pronunciarlo correctamente.

Esta es la sintaxis:

```
<say-as interpret-as="string" format="digit string" detail="string"> <say-as>
```

- El atributo `interpret-as` es obligatorio. Este define el tipo de contenido, y puede tomar cualquiera de los siguientes strings: `date`, `cardinal`, `ordinal`,`characters`, `time` y `telephone`.
- Los atributos `date`, `time` y `telephone` se supone que sean usados para especificar el tipo de contenido apropiado y un formato específico, pero al momento de escribir este artículo, estos 3 atributos son simplemente ignorados por el sintetizador de voz en Windows Phone 8..
- Los atributos `cardinal` y `ordinal` instruyen al sintetizador sobre la forma en que debe decir los números.
- El atributo `characters` instruye al sintetizador para leer el texto como caracteres individuales.

Aquí está un ejemplo:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"es\">";
ssmlText += "<p>Este es un número ordinal: <say-as interpret-as=\"ordinal\">121</say-as></p>";
ssmlText += "<p>Este es un número cardinal: <say-as interpret-as=\"cardinal\">121</say-as></p>";
ssmlText += "<p>Y estos son solo números individuales: <say-as interpret-as=\"characters\">121</say-as></p>";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

El elemento sub es usado para definir un sobrenombre para un string. Probablemente no es demasiado útil ya que funciona solo como un alias de una sola vez, así que probablemente su intención es solo dar mas claridad al documento SSML proveyendo una forma de tener una representación escrita y hablada del texto.

Esta es la sintaxis:

```
<sub alias="string"> </sub>
```

Y aquí hay un ejemplo:

```
ssmlText = "<speak version=\"1.0\" ";
ssmlText += "xmlns=\"http://www.w3.org/2001/10/synthesis\" xml:lang=\"es\">";
ssmlText += "Este código corre en <sub alias=\"Windows Phone 8\">WP8</sub>";
ssmlText += "</speak>";
await synth.SpeakSsmlAsync(ssmlText);
```

En este ejemplo las palabras "Windows Phone 8" son leídas en lugar de WP8.

Ejecutando un documento SSML externo

Los documentos SSML externos son archivos XML que siguen la estructura y sintaxis del lenguaje SSML que ha sido mostrada en todos los ejemplos anteriores.

Para ejecutar archivos SSML la clase `SpeechSynthesizer` provee el método `SpeakSsmlFromUriAsync`, el cual es muy similar a `SpeakSsmlAsync` y `SpeakTextAsync` pero éste recibe un objeto Uri como parámetro.

(C) Copyright Nokia 2013. All rights reserved.

Para este ejemplo, usarmos un archivo llamado **SSML1.xml**, el cual contiene el siguiente texto:

```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis" xml:lang="en-US">
  <voice gender="male" xml:lang="es">
    <prosody rate="0.8">
      <p>Gracias por leer el artículo, y gracias por probar los ejemplos</p>
      <p>Ahora sea creativo, y desarrolle aplicaciones asombrosas para esta fantástica plataforma</p>
      <voice gender="male" xml:lang="en-US">Bye</voice>
    </prosody>
  </voice>
</speak>
```

Luego de que el archivo ha sido creado (o ha sido agregado al proyecto), el paso final es llamar al método `SpeakSsmlFromUriAsync` para ejecutarlo:

```
await synth.SpeakSsmlFromUriAsync(new Uri("ms-appx:///Assets/SSML1.xml"));
```

Note: La documentación official de Microsoft sobre TTS para Windows Phone (link) usada para investigar y probar algunos de los temas incluídos en este artículo, incluye un ejemplo con un archivo SSML externo con extensión **ssml**, pero usar esa extension provoca una excepción `IO.FileNotFoundException`, mientras que archivos con extension **xml** funcionan bien.

Código Fuente

El proyecto con el código de todos los ejemplos incluídos en este artículo (version en Inglés) puede ser descargado aquí: AdvancedTTS test.zip

Material de referencia e información adicional

Más información sobre el lenguaje SSML

- Speech Synthesis Markup Language Reference
- Speech Synthesis Markup Language (SSML) Version 1.0 (W3C Recommendation)

Más información sobre archivos lexicon

- Pronunciation Lexicon Specification (PLS) Version 1.0

Más información sobre alfabetos

- Phonetic Alphabet Reference

Más información sobre text-to-speech para Windows Phone

- Text-to-speech (TTS) for Windows Phone

(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.