VoiceXML Tutorial

Part 1: VoiceXML Basics and Simple Forms
What is VoiceXML?

- XML Application
- W3C Standard
- Integration of Multiple Speech and Telephony Related Technologies
  - Automated Speech Recognition (ASR)
  - Text-to-Speech Synthesis (TTS)
  - DTMF
  - Interactive Voice Response (IVR)
- What to use it for
  - Describe Voice UIs for web content access over the phone
  - Implement Speech-enables telephony applications
VoiceXML Applications

- **Voice portals**
  - Provide personalized services to access information like stock quotes, weather, restaurant listings, news, etc.

- **Location-based services**
  - Receive targeted information specific to the location you are dialing from
  - Applications use the telephone number you are dialing from

- **Voice alerts (such as for advertising)**
  - Send targeted alerts to a user
  - The user would sign up to receive special alerts informing him of upcoming events.

- **Commerce**
  - Implement applications that allow users to purchase over the phone.
  - Mostly used for products that don't need a lot of description (such as tickets, CDs, office supplies, etc.)
The Big Picture
VoiceXML and XML

- Based on XML Tag/Attribute Format
  - Elements **must** be properly nested!

  ```xml
  <element attr_name="attr_value">
    ..contained elements..
  </element>
  ```

- All documents start with

  ```xml
  <?xml version="1.0"?>
  ```

- All other instructions are enclosed within the `<vxml>` tag, called "root element"

  ```xml
  <vxml version="2.0">
    ..VoiceXML Instructions..
  </vxml>
  ```
Example 1
Basic Document Layout

```xml
<?xml version="1.0"?>
<vxml version="2.0">

... VoiceXML Elements ...

</vxml>
```
VoiceXML Application Structure

- VoiceXML documents may be grouped together into applications
  - A root document serves as the initial point of the application
  - Leaf documents are connected to it
- Each document is made up from a set of dialogs
- A session begins as soon as the user begins interacting with a VoiceXML document
Dialogs

- **Dialogs** are the building blocks for VoiceXML documents
  - Collect data from the user
  - Offer choices to the user
- The user is **always** in a dialog
  - Each dialog is followed by a new one
  - If no dialog is specified next, the application ends
- Two top-level dialog elements
  - Forms: `<form>`
  - Menus: `<menu>`
  - The difference is semantic. The behavior of a `<menu>` can easily be implemented with a `<form>`
Forms

- Discrete dialog elements
  - Denoted by `<form>` tag
  - Optional “id” attribute to specify name
  - Responsible for executing some part of the dialog

- Forms contain various elements
  - Elements perform tasks required by form
  - Several, among which those known as “form items”

```html
<form id="welcome">
  ..elements..
</form>
```
Example 2
VoiceXML Document with Form

```xml
<?xml version="1.0"?>
<xml version="2.0">
<form id="welcome">
... Form Contents ...
</form>
</xml>
```
Delivering Content

- Two content delivery methods
  - Text-to-Speech
  - Pre-Recorded Prompts

- `<prompt>`
  - All text is actually a TTS prompt
  - Enclosure in `<prompt>` tags provides extra control over its behavior.

- `<audio>`
  - Typically an 8kHz, 8-bit monaural audio file
Example 3
Hello World!

Remarks on `<vxml>`
- **base**
  Base URI
- **lang**
  Language and Locale type
- **application**
  URI of app root document (only for leaf docs)
- **version**
  VoiceXML version

Output:
- “Hello World!”
Example 4
Prompts

- **bargain**: the ability to interrupt a prompt
- **cond**: the condition that allows the prompt to be played
- **timeout**: the time the user has to respond to the prompt without causing an error

```xml
<?xml version="1.0"?>
<xml version="1.0">
  <form id="welcome">
    <block>
      <prompt cond="(new Date()).getDay() == 0" bargain="false">
        Sorry! We are closed on Sunday.
      </prompt>
      <prompt cond="(new Date()).getDay() != 0" bargain="false">
        Hello! Thanks for calling.
      </prompt>
    </block>
  </form>
</xml>
Example 5
Pre-Recorded Audio Playback

```xml
<?xml version="1.0"?>
<vxml version="2.0">
  <form id="welcome">
    <block>
      <prompt cond="(new Date()).getDay() != 0" bargein="false">
        Hello! Thanks for calling.
        <audio src="welcome.wav" caching="fast" fetchhint="prefetch"/>
      </prompt>
    </block>
  </form>
</vxml>
```

- **src**: the file containing the prompt
- **fetchhint**: when should the browser download this file (prefetch or safe)
- **caching**: how to manage the cache
Example 6
Simple Dialog

- Output
  - “Hello World! My name is Chris.”

```xml
<?xml version="1.0"?>
<vxml application="intro.vxml" version="2.0">
  <form id="welcome">
    <block>
      <prompt>
        Hello World!
      </prompt>
    </block>
  </form>
  <form id="name">
    <block>
      My name is Chris.
    </block>
  </form>
</vxml>
```
Collecting User Feedback

- **Grammars**
  - A list of permissible vocabulary for the user to select from
  - Two Grammar types
    - Speech Grammars
      - Legal Utterances
    - DTMF Grammars
      - Keypad sequences

- **Events**
  - Thrown by the application for certain reasons
    - No user input, erroneous input, user request for help
Basic Form Items

- **Field Items:** Gather information from caller to fill variables (*field item variables*)
  
  | `<field>` | input from user via speech or DTMF, as defined by a grammar |

- **Control Items:** Enclose non-recognition based tasks
  
  | `<block>` | encloses sequence of statements for prompting and computation |
Example 7
Gathering User Input with <field>

```
<form id="getPhone">
  <field name="PhoneNo">
    <prompt>
      What's your phone number?
    </prompt>
    <grammar src="../grammars/phone.gram" type="application/x-jsgf" />
    <help>
      Please say your ten digit phone number.
    </help>
  </field>
</form>
```

- Prompt user for input
- Recognize input according to specified grammar (type="phone" utilizes the built-in phone number grammar)
- Catch any events appropriate to portion of dialog
- Fill field variable "PhoneNo" with recognized user response
Form Item Variables

- Associated with every form item
- Defined by `name` attribute
  - Default value: `undefined`
  - Gets filled in with user collected result

Guard Conditions
- Default: test whether variable has a value
  - If undefined, form item is executed
  - If defined, form item is skipped
Gathering Data with a Form

<?xml version="1.0"?>
<vxml version="2.0">
<form id="welcome">
  <field name="selection">
    <prompt>
      Please choose News, Weather or Sports.
    </prompt>
    <grammar>
      [ news weather sports ]
    </grammar>
  </field>
  <block>
    <goto next="choose.vxml"/>
  </block>
</form>
</vxml>