# Introduction to VoiceXML

- what it is, why use it
- how to get started
- sample code

# What is VoiceXML?

**Consider a Web site:**

- HTML page on a web server
- web developers don't have to deal with setting up a web server. They just make HTML.

**The goal is to have the same with voice apps:**

- VoiceXML pages on a VoiceXML platform
- developers don't have to deal with integrating speech recognition, XML parsing, text-to-speech, and telephony. Just focus on dialogue flow.

# Why use VoiceXML?

**Best for:**

- Finite State-based dialogue apps
  - FS methods can get you pretty far
  - industry likes them
- IVR (interactive voice response) telephone-based applications

**Not so Good for:**

- non-Finite State dialogue management. i.e. most of the things you're learning in this class!
- things that need custom SR and TTS
  - performance of these can be improved dramatically with custom language, acoustic, and voice models. (of course you could develop your own VoiceXML platform for this.)

# How to Get Started

**Build your own platform:**

- speech recognizer (e.g. Sphinx, Sonic)
- VoiceXML parser (e.g. OpenVXI)
- text-to-speech generator (e.g. Festival, FreeTTS)
- PSTN (telephone network) link - optional?

**Or, find a service provider:**

- start here: http://www.kenrehor.com/voicexml/#vsps
- find one with free developer accounts (I used VoiceGenie last year)
- typical set-up:
  - you specify the URL of a web-served .vxml file
  - you call a phone number, enter extension
  - the provider accesses and runs the .vxml file

# Sample .vxml

**Hello World:**

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <block>
      <prompt>
        Hello world!
      </prompt>
    </block>
  </form>
</vxml>
```

(from http://en.wikipedia.org/wiki/VoiceXML)

# Sample .vxml

**Simple Choice:**

```
<?xml version="1.0"?>
<vxml version="2.0">
<menu>
  <prompt>
    Say one of: <enumerate/>
  </prompt>
  <choice next="http://www.sports.example/start.vxml">
     Sports
  </choice>
  <choice next="http://www.weather.example/intro.vxml">
     Weather
  </choice>
  <choice next="http://www.news.example/news.vxml">
     News
  </choice>
  <noinput>Please say one of <enumerate/></noinput>
</menu>
</vxml>
```

(from http://www.w3.org/Voice/Guide/)

# Sample .vxml

**Dynamically Generated:**

```
#!/usr/bin/perl

open I, "<data.txt";
while (<I>) {
        chomp;
        $message .= $_;
}
close I;

print qq*
<?xml version="1.0"?>
<vxml version="2.0">
   <form>
      <block>
        The message of the day is: $message
        <goto next="http://yoursite.example/next.vxml"/>
      </block>
   </form>
</vxml>
*;
```

(from my homework last year)

# Sample .vxml

**Idea: send info as part of URL and use Perl CGI 'GET'**

```
...
<if cond="choice=='cheese'">
      <goto next="http://yoursite.example/pizza.pl?session=22&topping=cheese"/>
<elseif cond="choice=='mushrooms'"/>
      <goto next="http://yoursite.example/pizza.pl?session=22&topping=mushrooms"/>
</if>
...
```

You could dynamically write session information as part of URL.

VoiceXML can interact with databases or server-side programs written in any language, as long as they return .vxml

You could thus have sessions of arbitrary complexity. This is stretching the idea of VoiceXML, though.

# Summary

- Voice XML is meant to let you focus on dialogue flow

- It's most intuitive when used with Finite State methods

- You can either develop your own platform or find a provider

- It can be used to do quite a bit.