Intorduction to VoiceXML (4)

Grammar

Nuance Grammar Specification Language (GSL)

Learning Objectives

- A Simple example
- Understanding '?'
- Understanding '*'
- Understanding '+'
- Allowing DTMF input in your grammars
- Specifying both voice and DTMF grammars
- Integrating grammars into VoiceXML

grammar.vxml

```
<vxml version="2.0">
<form>
   <field name="command">
   <grammar type="application/x-gsl" mode="voice">
   <![CDATA[
   [ [bail (drop me)] {<command "drop">}
      [(enroll me) (sign me up)] {<command "enroll">}
   ]
   ]]>
   </grammar>

   <prompt>Add or drop?</prompt>

   <filled>
      <audio>You said <value expr="command"/></audio>
   </filled>
   </field>
</form>
</vxml>
```

grammar1.vxml

```
<vxml version="2.0">
<form>
   <field name="command">
   <grammar type="application/x-gsl" mode="voice">
   <!-- TellMe Studio ScratchPad cannot handle this file.
        Use Application URL can run correctly. -->
   <![CDATA[
   [ [good] {<reply "good">}
      [terrible] {<reply "bad">}
   ]
   ]]>
   </grammar>

   <prompt>How do you feel about the movie?</prompt>

   <filled>
      You said <value expr="command"/>.
   </filled>
   </field>
</form>
</vxml>
```

grammar1.vxml

- Computer: How do you feel about the movie?
- User: Good
- Computer: You said “Good”.

- Computer: How do you feel about the movie?
- User: Terrible.
- Computer: You said “Bad”.

- Computer: How do you feel about the movie?
- User: Very good.
- Computer: Sorry. An error has occurred.

grammar2.vxml

```
<vxml version="2.0">
<form>
   <field name="command">
   <grammar type="application/x-gsl" mode="voice">
   <![CDATA[
   [ [good ok] {<reply "good">}
      [bad terrible] {<reply "bad">}
   ]
   ]]>
   </grammar>

   <prompt>How do you feel about the movie?</prompt>

   <filled>
      You said <value expr="command"/>.
      <clear/>
      <reprompt/>
   </filled>
   </field>
</form>
</vxml>
```

grammar2.vxml

- Computer: How do you feel about the movie?
- User: Good
- Computer: You said “Good”. How do you feel about the movie?
- User: OK.
- Computer: You said “Good”. How do you feel about the movie?
- User: Terrible.
- Computer: You said “Bad”. How do you feel about the movie?
- User: Very good.
- Computer: Sorry. An error has occurred.

grammar3.vxml

```
<vxml version="2.0">
<form>
   <field name="command">
   <grammar type="application/x-gsl" mode="voice">
   <![CDATA[
    [ [very good] {<reply "good">}
      [bad] {<reply "bad">}
   ]
   ]]>
   </grammar>

   <prompt>How do you feel about the movie?</prompt>

   <filled>
      You said <value expr="command"/>.
      <clear/>
      <reprompt>
   </filled>
   </field>
</form>
</vxml>
```

grammar3.vxml

- Computer: How do you feel about the movie?
- User: Good
- Computer: You said “Good”. How do you feel about the movie?
- User: Very good.
- Computer: You said “Good”. How do you feel about the movie?
- User: Very
- Computer: You said “Good”. How do you feel about the movie?

grammar4.vxml

```
<vxml version="2.0">
<form>
   <field name="command">
   <grammar type="application/x-gsl" mode="voice">
   <![CDATA[
   [ [(very good) ok] {<reply "good">}
      [bad] {<reply "bad">}
   ]
   ]]>
   </grammar>

   <prompt>How do you feel about the movie?</prompt>
   <nomatch>Please try again.</nomatch>
   <filled>
      You said <value expr="command"/>.
      <clear/>
      <reprompt>
   </filled>
   </field>
</form>
</vxml>
```

grammar4.vxml

- Computer: How do you feel about the movie?
- User: Ok.
- Computer: You said “Good”. How do you feel about the movie?
- User: Very good.
- Computer: You said “Good”. How do you feel about the movie?
- User: Very
- Computer: Please try again. How do you feel about the movie?
- User: Good
- Computer: Please try again. How do you feel about the movie?

grammar5.vxml

```
<vxml version="2.0">
<form>
   <field name="command">
   <grammar type="application/x-gsl" mode="voice">
   <![CDATA[
     [ [(?very good)] {<reply "good">}
     ]
   ]]>
   </grammar>

   <prompt>How do you feel about the movie?</prompt>
   <nomatch>I don't understand. Please try again.</nomatch>
   <filled>
      You said <value expr="command"/>.
      <clear/>
      <reprompt>
   </filled>
   </field>
</form>
</vxml>
```

grammar5.vxml

- Computer: How do you feel about the movie?
- User: Good.
- Computer: You said “Good”. How do you feel about the movie?
- User: Very good.
- Computer: You said “Good”. How do you feel about the movie?
- User: Very
- Computer: I don’t understand. Please try again.

grammar6.vxml

```
<vxml version="2.0">
<form>
   <field name="command">
   <grammar type="application/x-gsl" mode="voice">
   <![CDATA[
     [ [(*very good)] {<reply "good">}
     ]
   ]]>
   </grammar>

   <prompt>How do you feel about the movie?</prompt>
   <nomatch>I don't understand. Please try again.</nomatch>
   <filled>
      You said <value expr="command"/>.
      <clear/>
      <reprompt>
   </filled>
   </field>
</form>
</vxml>
```

grammar6.vxml

- Computer: How do you feel about the movie?
- User: Good.
- Computer: You said “Good”. How do you feel about the movie?
- User: Very good.
- Computer: You said “Good”. How do you feel about the movie?
- User: Very very very very very good.
- Computer: You said “Good”. How do you feel about the movie?
- User: Terrible.
- Computer: I don’t understand. Please try again.

- My experiments show that “*” behaves the same as “?”. How are yours. I wish you to find out an example to distinguish them.

grammar7.vxml

```
<vxml version="2.0">
<form>
   <field name="command">
   <grammar type="application/x-gsl" mode="voice">
   <![CDATA[
     [ [(+very good)] {<reply "good">}
     ]
   ]]>
   </grammar>

   <prompt>How do you feel about the movie?</prompt>
   <nomatch>I don't understand. Please try again.</nomatch>
   <filled>
      You said <value expr="command"/>.
      <clear/>
      <reprompt>
   </filled>
   </field>
</form>
</vxml>
```

grammar7.vxml

- Computer: How do you feel about the movie?
- User: Good.
- Computer: I don't understand. Please try again.
- User: Very good.
- Computer: You said "Good". How do you feel about the movie?
- User: Very very very very very good.
- Computer: You said "Good". How do you feel about the movie?
- User: Terrible.
- Computer: I don't understand. Please try again.

DTMF

- Both DTMF and voice inputs are allowed.

```
<grammar type="application/x-gsl" mode="voice">
<![CDATA[
  [ [sales] {<dept "010">}
    [marketing] {<dept "020">}
    [engineering] {<dept "030">}
    [(public relations) (p r)] {<dept "040">}
  ]
]]> </grammar>

<grammar type="application/x-gsl" mode="dtmf">
<![CDATA[
  [ [dtmf-1] {<dept "010">}
    [dtmf-2] {<dept "020">}
    [dtmf-3] {<dept "030">}
    [dtmf-4] {<dept "040">}
  ]
]]> </grammar>
```

- The pound key (#) is typically used to allow the user to signify the end of DTMF grammar input.

Specifying both voice and DTMF grammars

- If you do not set the mode attribute on the grammar element, both voice and DTMF are supported.

```
<grammar type="application/x-gsl">
<![CDATA[
  [ [sales dtmf-1] {<dept "010">}
    [marketing dtmf-2] {<dept "020">}
    [engineering dtmf-3] {<dept "030">}
    [(public relations) (p r) dtmf-4] {<dept "040">}
  ]
]]> </grammar>
```

- Although GSL supports the specification of voice and DTMF in the same grammar, it is recommended that you split voice and DTMF into separate grammars to maximize performance.

DTMF.vxml

- Computer: Please tell me the department name.
- User: Sales:
- Computer: You said “010”. Please tell me the department name.
- User: (press 2)
- Computer: You said “020”. Please tell me the department name.
- User: P R
- Computer: You said “040”. Please tell me the department name.

Integrating grammars into VoiceXML

- Because the GSL syntax uses characters that are reserved by XML ("<", ">"), in-line grammars must be protected from the XML parser within a CDATA section.
- grammars can also be defined in an external document and pulled into the VoiceXML document using the src attribute of the grammar element.
- Using the src attribute provides the following advantages:
  - grammars can be generated dynamically (e.g. via PHP or ASP)
  - grammars can be reused by multiple dialogs or applications
  - grammars can be updated or localized without modifying the application source code

grammar10.vxml

```
<vxml version="2.0">
<form>
   <field name="decision">
   <grammar type="application/x-gsl" src="yesno.gsl">
   </grammar>

   <prompt>Please tell me your decision.</prompt>

   <catch event="noinput nomatch">
      Please say Yes or No.
      <break size="medium"/>
      <reprompt/>
   </catch>

   <filled>
      <audio>You said <value expr="decision"/></audio>
   </filled>
   </field>
</form>
</vxml>
```

yesno.gsl

```
[
  [yes sure yeah okay check affirmative (ten four) dtmf-1]
    {<conf "yes">}
  [no nope negative dtmf-2]
    {<conf "no">}
]

    ; There is no implicit grammar for DTMF.
    ; You need to specify it explicitly.
```

Summary of Syntax

- A comment begins after a semi-colon and terminates at the end of the line.
  - Multi-line comments are not supported in GSL All the words within the grammar are in lowercase.
- Uppercase letters are reserved to define the name and to reference subgrammars.
- Square brackets define an "or" condition.
- Parenthesis define an "and" condition.
- Use "?" before a word or phrase when that word or phrase may occur zero or one time
- Use “*" before a word or phrase when that word or phrase may occur zero or more times
- Use “+" before a word or phrase when that word or phrase may occur one or more times

Oral Presentation

- May 5th, Friday.
- Upload your PowerCam™ presentation in advance.
- Presentation Scoring
  - Correctness 80%
  - Related Reference 10%
  - Examples 10%