



Intorduction to VoiceXML

Outline

- XML
- VoiceXML
- Building your VoiceXML application on TellMe Studio

XML

- Extensible Markup Language
- The de facto standard for defining structured documents and data on the Web
- Many extensions:
 - ebXML - Electronic Business XML
 - ccXML - Call Control eXtensible Markup Language
 - VoiceXML

Understanding XML Documents

- An XML **document** is comprised of one or more named elements organized into a nested hierarchy.
- An **element** is an opening tag, some data, and a closing tag.
- A **tag** is an element name preceded by a less-than symbol (<) and followed by a greater-than (>) symbol.

- For any given element, the name of the opening tag must match that of the closing tag.
- A closing tag is identical to an opening tag except that the less-than symbol is immediately followed by a forward-slash (/).
- Tag names are **case-sensitive**.
 - ☞ Although the following examples employ lowercase, either case may be used as long as the opening and closing tag names are consistent.
 - `<welcome>Welcome to Tellme University</welcome>`
- If an element does not contain any data, the opening and closing tags can be combined. Observe the location of the forward slash just prior to the greater-than (>) symbol.
 - `<disconnect/>`

Attributes

- An element may include zero or more attributes.
- An **attribute** specifies properties of the element that you modify and consists of a name/value pair.
- Attribute values must be contained in matching single or double quotes.

```
<salutations>
```

```
  <welcome type="texan">Howdy, partner</welcome>
```

```
  <welcome type="australian">G'day, mate</welcome>
```

```
</salutations>
```

Root Element

- Elements may be arranged in an infinitely nested hierarchy, but only one element in the document can be designated as the root document element.
- The **root document** element is the first element that appears in the document.
- With the exception of the comment element, all other elements in the document must be children of the root element.
 - In the following example, `<customer>` is the root element and it has three children. The child element `<address>` has four children.

```
<customer id="C1234">
  <lname>Smith</lname>
  <fname>John</fname>
  <address type="biz">
    <street>1310 Villa Street</street>
    <city>Mountain View</city>
    <state>CA</state>
    <zip>94041</zip>
  </address>
</customer>
```

Comments

- It's always a good idea to document your work so that other programmers can understand it.
- A comment begins with the combination of characters "<!--" and ends with the combination of characters "-->".
- Comments may appear as a child of any element in an XML document.
- They can also appear before or after the root element. A comment may span multiple lines but cannot be nested.

```
<assembly>  
  <part>A451</part> <!-- headset -->  
  <part>C200</part> <!-- crank -->  
  <part>X999</part> <!-- derailleur -->  
</assembly>
```

XML Validation

- An XML schema can be formally defined using a Document Type Definition (DTD).
- The DTD defines the elements and attributes that are allowed in the document.

☞ The following DTD formally describes the schema for the customer example above.

```
<!ELEMENT customer (lname, fname, address+)>
<!ATTLIST customer id ID #REQUIRED>
<!ELEMENT lname (#PCDATA)>
<!ELEMENT fname (#PCDATA)>
<!ELEMENT address (street, city, state, zip)>
<!ATTLIST address type (biz | home) #REQUIRED>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
```


XML Validation (cont.)

- An XML parser can typically be configured to validate an XML document against its corresponding DTD.
- A DTD is associated with an XML document via a DOCTYPE declaration.
- A DOCTYPE declaration can appear at the top of an XML document **before the root element**.

☞ In the following declaration, the document containing the root element `<customer>` is validated against a DTD located in the file "cust.dtd."

```
<!DOCTYPE customer SYSTEM "cust.dtd">
```

Reserved characters

- less-than (<), greater than (>), and ampersand (&)
- To express these characters in your document data, use the equivalent character entity:

less-than (<)	<
greater than (>)	>
ampersand (&)	&

- `<equation> x < 2 </equation>`
- `<rhyme> Jack & Jill </rhyme>`

Comparing HTML and XML

XML	HTML
Can be used to define an arbitrary vocabulary for any problem domain.	Defines a specific vocabulary used to create visually rendered documents.
All tags must be closed.	Some tags, such as IMG and INPUT, need not be closed.
All attributes must be associated with a value.	Some attributes don't require an explicit value (e.g., TABLE BORDER).
All attribute values must be surrounded by quotes.	Attribute values need only be quoted when containing embedded spaces.
Entities beyond &, <, and > must be explicitly declared via ENTITY declarations in the DTD.	Most entities are implicitly defined by the parser.
Parser is generally unforgiving. Basic rules about tags and attributes are enforced rigorously and DTD, if supplied, is followed to the letter.	Parser is extremely forgiving and forgotten tags are implicitly inserted into the document.
When a DTD is supplied, unrecognized tags are rejected and the parse fails.	Unrecognized tags are ignored.
May contain only a single root element.	Should contain only a single root element (HTML), but the parser may be forgiving if the HTML element is omitted.

VoiceXML 2.0

- VoiceXML is a specific kind of XML designed to describe voice applications.
 - <http://www.w3.org/TR/voicexml20/>
- Whereas XML is designed to represent **arbitrary data**, VoiceXML describes grammars, prompts, event handlers, and other data structures useful in describing **voice interaction between a human and a computer**.

Application Structure (cont.)

- At the root of every VoiceXML document is a root element, the vxml element.
- This element should contain one or more elements representing dialogs.
- VoiceXML 2.x provides two types of dialogs: form and menu.
 - While **menu** is a convenient shorthand for certain situations, VoiceXML authors will spend most of their time writing **form** elements; thus, we shall focus on the **form** element.

<Form> Element

- A **form** with a number of elements that either provide information to the user or request input from the user.
- Every form contains one or more form items.
- The first form item that you'll examine is block.
- A **block** is a container for "executable content".
 - It contains commands (i.e. VoiceXML elements) that are executed sequentially.
 - **block** elements are often used for presenting information to the user.
 - For example, if you place text within a block, it's treated as a command to queue audio that will ultimately be played to the user.

Example Code

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">  
  <form>  
    <block>Hello, world!</block>  
  </form>  
</vxml>
```

- This document is the simplest functional VoiceXML document.
 - It consists of a **vxml** element,
 - which contains a single dialog (a **form**),
 - which contains a single form item (a **block**),
 - which contains a single statement

Application Structure

- A VoiceXML application consists of a set of VoiceXML documents,
- Each VoiceXML document contains one or more dialogs describing a specific interaction with the user.
- Dialogs may
 - present the user with information or
 - prompt the user to provide information,
- when complete, they can redirect the flow of control
 - to another dialog in that document,
 - to a dialog in another document in the same application, or
 - to a dialog in another application entirely.

Jumping between dialogs with <goto>

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">  
  <form>  
    <block>  
      Goodbye, world!  
      <goto next="document2.vxml"/>  
    </block>  
  </form>  
</vxml>
```

<goto>

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">  
  <form>  
    <block>  
      Goodbye, world!  
      <goto next="http://solomon.ipv6.club.tw/Lang/VoiceXML/hello.vxml"/>  
    </block>  
  </form>  
</vxml>
```

Jump to another form in the same document

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">  
  <form>  
    <block> To infinity  
    <goto next="#infinity"/>  
  </block>  
</form>  
  
  <form id="infinity">  
    <block> and beyond  
    <exit/>  
  </block>  
</form>  
</vxml>
```

Collecting user input

- The primary method of gathering user input is via the field element
- **field** is a special kind of form item
 - ☞ A **field** is a blank slate waiting to be filled by user input, whereas a **block** is capable only of executing instructions such as generating audio and **goto** elements.
- For each **field**, you must provide a grammar, which is a piece of information describing the allowable user input for a given field.
- You can specify grammars in one of two ways:
 - You can set the *type* attribute of the **field** element. This causes the VoiceXML interpreter to use one of its "built-in" grammars.
 - You can use one or more **grammar** elements. A **grammar** can contain arbitrary words and phrases expressed in a special grammar description language.

Collecting user input (cont.)

- A **field** element has several important components:
 - a name,
 - one or more prompts,
 - one or more grammars, and
 - instructions to be executed when the field has been filled.
- The *name* attribute of the **field** identifies the variable associated with the user input you collect.
- You use one or more prompt elements to instruct the user on how to fill the form item variable.
- You use a filled element to provide instructions to be executed when the **field** element's form item variable has been filled.

Example

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <field name="famous" type="boolean">
      <prompt> would you like to be famous?
    </prompt>

    <filled> got it!
    <if cond="famous">
      let's schedule an audition
      <goto next="schedule.vxml" />
    <else /> infamous is a reasonable alternative.
      <goto next="infamous.vxml" />
    </if>
    </filled>
  </field>
</form>
</vxml>
```

Handling the unexpected

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">
```

```
<form>
```

```
<field name="famous" type="boolean">
```

```
<prompt>
```

```
would you like to be famous?
```

```
</prompt>
```

```
<noinput>
```

```
Sorry I didn't hear you.
```

```
For famous press 1.
```

```
For infamous press 2.
```

```
</noinput>
```

```
<nomatch>
```

```
Sorry I didn't get that.
```

```
To be famous say yes.
```

```
Otherwise, say no.
```

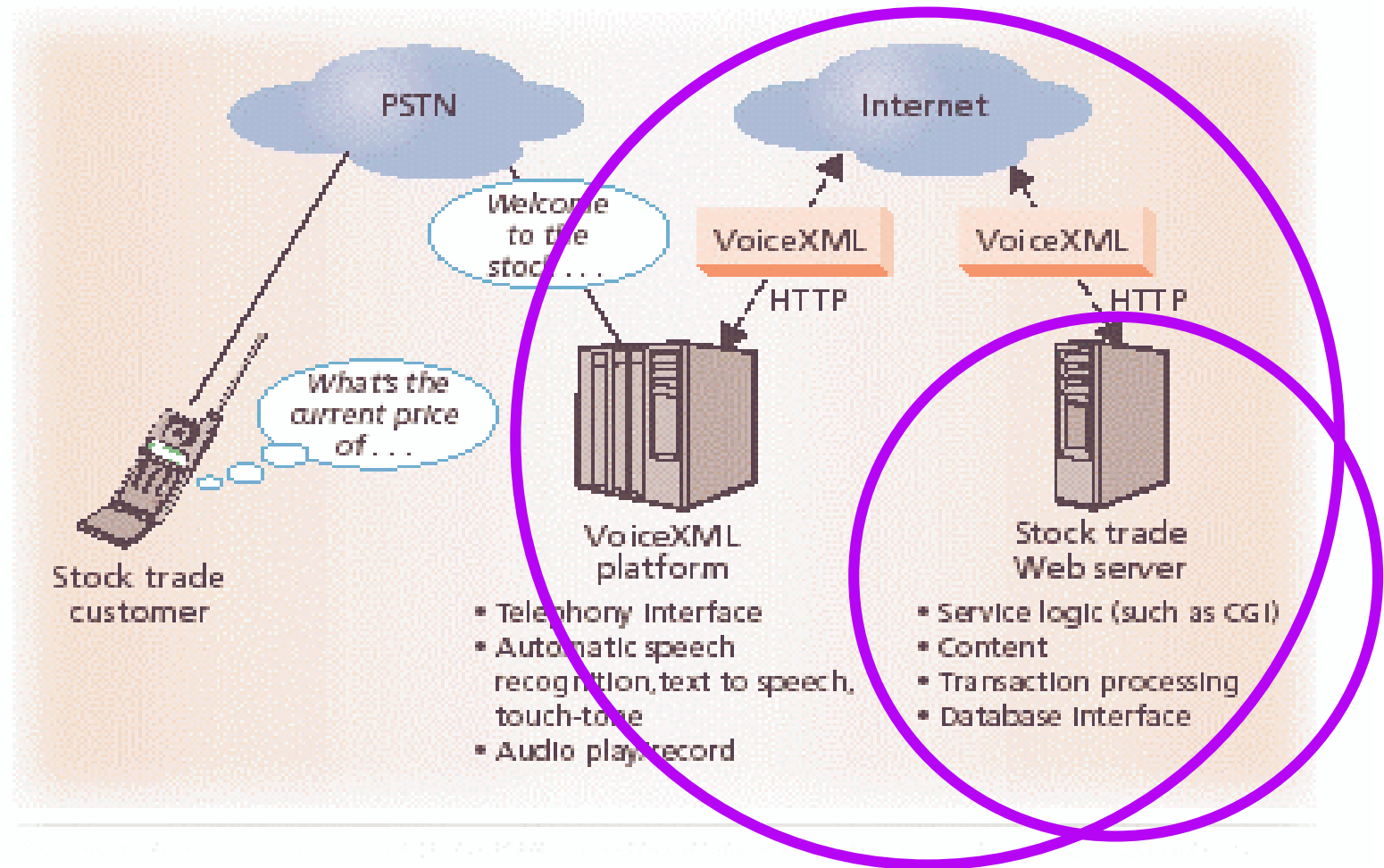
```
</nomatch>
```



```
<filled>  
  got it!  
  <if cond="famous">  
    let's schedule an audition  
  <else />  
    infamous is a reasonable  
    alternative.  
  </if>  
  <clear/>  
  </filled>  
</field>  
</form>  
</vxml>
```


The clear element clears
the value of any form
item variables - in this
case the variable
"famous"

VoiceXML Conceptual Architecture



TellMe Studio

- <http://studio.tellme.com/>
- sip:8005558965@sip.studio.tellme.com



The screenshot shows the TellMe Studio web interface. At the top, there are two tabs: "Application URL" and "Scratchpad". Below the tabs is a toolbar with icons for "debug log", "check syntax", "grammar tools", "VoiceXML terminal", "record by phone", and "edit my preferences". The main content area contains the following text:

Type some VoiceXML below, call **1-800-555-VXML**, and enter Developer ID **975053** to preview it. [\[International | VoIP^{new}\]](#)

✓ Your change was successful. The syntax checker was run. (No errors were detected in your VoiceXML.)

VoiceXML Scratchpad name: [what's this?](#)

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <block>
      Hello, World.
    </block>
  </form>
</vxml>
```



Application URL

The screenshot shows a web interface with two tabs: "Application URL" (active) and "Scratchpad". The interface includes a toolbar with icons for "debug log", "check syntax", "grammar tools", "VoiceXML terminal", "record by phone", and "edit my preferences". The main content area contains the following text:

Enter the URL to your VoiceXML below, call **1-800-555-VXML**, and enter Developer ID **975053** to preview it.
[[International](#) | [VoIP](#) ^{new}]

✔ Your change was successful. The syntax checker was run. ([No errors](#) were detected in your VoiceXML.)

Application URL [what's this?](#)

Set URL to a previous value

Did You Know?
Looking for a great book on VoiceXML 2.0? Get [VoiceXML: Professional Developer's Guide](#).

a.php

```
<vxml version="2.1" xmlns="http://www.w3.org/2001/vxml">
```

```
<form>
```

```
<block>
```

```
<?php
```

```
for ($i=1; $i<=10; $i++)
```

```
    echo $i . "\n";
```

```
?>
```

```
</block>
```

```
</form>
```

```
</vxml>
```