Introduction to XML – the Language of Web Services

Tony Obermeit
Senior Development Manager, Wed ADI Group
Oracle Corporation

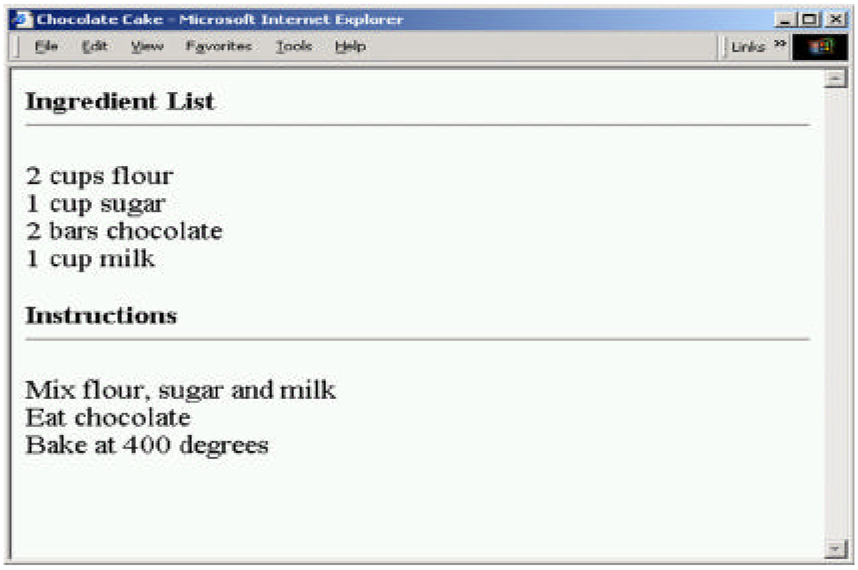
Introduction to XML – In this presentation, we will be discussing:
1) The origins of XML - what lead to the creation of the xml la nguage? We won't be focusing on the details of the standard or the standards body but will focus on actual physical examples of xml
2) The structure of an XML document
3) 10 Rules of a "Well Formed Document"
4) Validating the document by either a DTD or an XML-Schema specification
5) XML Parsers and how XML is processed programmatically
6) What about the "Presentation" of the Data?
7) The use of XML in the standards for Web Services

1) Origin of XML

- HTML fuelled growth of internet
- However, significant problems in HTML
- NOT ext ensible
- TOO display centric
- ONLY 1 view of the data
- LITTLE semantic structure (no meaning)
- Examples following… We have an example of a web page and show how the same page can be coded in more than one way, totally different html, visually looks the same. To keep the presentation brief, we have only 2 examples of different code that looks the same, in practice I was able to code the html in at least 4 different ways that all visually looked the same.

HTML Recipe

Here is an html page, simple, easy to read. Next will be two examples of html code, both examples of code are significantly different but both produce the same presentation as we see here:


Chocolate Cake - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links

Ingredient List

2 cups flour
1 cup sugar
2 bars chocolate
1 cup milk

Instructions

Mix flour, sugar and milk
Eat chocolate
Bake at 400 degrees

HTML – 1st Example

Ingredients and Instructions marked by <b> Bold tag, each ingredient / instruction distinguishable only by the new line <br /> Break tag.

```
<html><head><title>Chocolate Cake</title><body>
<b>Ingredient List</b><hr />
<br />2 cups flour
<br />1 cup sugar
<br />2 bars chocolate
<br />1 cup milk
<br /><br /><b>Instructions</b>
<hr /><br />Mix flour, sugar and milk
<br />Eat chocolate
<br />Bake at 400 degrees
</body></html>
```

HTML – 2nd Example

Here, a different tag <dt> separates each ingredient / instruction.

```
<html><head><title>Chocolate Cake</title><body>
<b>Ingredient List</b><hr /><dl>
<dt>2 cups flour
<dt>1 cup sugar
<dt>2 bars chocolate
<dt>1 cup milk
</dl><br /><b>Instructions</b>
<hr /><dl>
<dt>Mix flour, sugar and milk
<dt>Eat chocolate
<dt>Bake at 400 degrees
</dl></body></html>
```

It is hard, if not impossible to know which tag will be used by a html page. One of the main problems with html is that it contains both the data and how the data should be displayed, ie the presentation.

Another problem with html is that it is hard to parse programmatically, the browsers are very forgiving which is helpful but that makes it real hard to parse, different html parsers forgive in different ways. For web automation (read services/e-commerce) the data needs to be readable and processable programmatically, not just visually.

Next let's look at a recipe in XML.

2) XML Document Structure

Text file containing Elements, Attributes & Text

```
<?xml version="1.0" ?>
<Recipe name="Chocolate Cake"  type="Desert" >
    <IngredientList>
        <Ingredient>2 cups flour</Ingredient>
        <Ingredient>1 cup sugar</Ingredient>
    </IngredientList>
    <Instruction>Sift the flour</Instruction>
</Recipe>
```

XML is just the data, no presentation instructions. Later we'll see how the presentation of the data is taken care of. (Note the **elements**, the **attributes** and the **text,** and note the hierarchical / tree structure of the document.)

3) 10 Rules – Well Formed XML

In order for XML to be used, it has to be "WELL FORMED". This means it is in a structure that makes it readable by an xml program.

1. **Must start with XML declaration**

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

2. **Must be only one document element**
    This highlights also the hierarchical / tree structure of all xml documents.

Valid Example(s)	Invalid Example
<?xml version="1.0" ?>	<?xml version="1.0"?>
<recipe>	<recipe>
</recipe>	</recipe>
or	<recipe>
<recipeBook>	</recipe>
<recipe></recipe>	
<recipe></recipe>	
</recipeBook>	

3. **Match opening & closing tags**
    HTML doesn't care if you close your tags, or the order in which you close them. XML is much fussier, you must close ALL tags, and you must close them in the hierarchical order.
    - Carry over from html origins
    - <hr> <p> or <bold><italic></bold></italic>
    - Browsers forgive, XML Parsers do NOT
    - <p></p> or <br />
    - <bold><italic></italic></bold>
    - <recipe></recipe>
    - XHTML – HTML using XML rules

4. **Comments allowed, but not inside attribute or element tag**

   Note, comments can also extend across multiple lines.

   ```
   <!-- Isn’t XML really cool? -->

   <!– So is being an Oracle User Group member!!! -->

   <!-- Tony for Prime Minister! ! !
            Tony for President! ! !
      -->
   ```

5. **Elements and Attributes must start with a letter**

   <Recipe> OK

   <Second third=“false”> OK

   <2nd> INVALID

   <Recipe 2nd=“true”> INVALID

6. **Attributes must go in the opening tag**

   Valid:

   ```
   <recipe name=“Chocolate Cake”
             category=“Desert”></recipe>
   ```

   Invalid:

   ```
   <recipe></recipe name=“Chocolate Cake”>
   ```

7. **Attributes must be enclosed in matching quotes**

   Can use either single or double quotes but must use same type to start and end attribute value

   Name=“Oracle User Group”

   Name=‘Oracle User Group’

8. **Only simple text for attributes, no nested values.**

   Nesting is allowed in elements, not in attributes.

9. **Use &lt; &amp; &gt; &quot; and &apos; for special characters. <   &   >   “   ‘**

10. **Write empty elements using <recipe /> syntax if no nested values, can still have attributes in tag <recipe type=“desert” />.**

With these 10 rules, we have a "Well Formed" xml document

- It means the xml can be read, processed or parsed. All XML must be well formed, if not well formed, can't be treated as XML, and therefore cannot be processed / parsed.
- 2nd part of xml basics is validating the xml document. This is optional and allows us to ensure the document follows a specific structure.
- Doesn't mean the structure makes sense.

```
<recipe model="Holden">
  <chapter></chapter>
  <engine cylinders="4"></engine>
<recipe>
```

- Well formed, but possibly meaningless

4) DTD – Document Type Definition

- Allows us to define the exact elements and attributes for the document
- These effectively become the rules of our own markup language, the extensible part of xml
- DTD – really only defines the structure, limited in what you can validate in regards to the text values of the element or attribute.

Recipe DTD

```
<!ELEMENT Recipe (Name, Description?, Ingredients?, Instructions?)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Ingredient (Qty, Item)>
<!ELEMENT Qty (#PCDATA)>
<!ATTLIST Qty unit CDATA #REQUIRED>
<!ELEMENT Item (#PCDATA)>
<!ATTLIST Item optional CDATA "0" isVegetarian CDATA "true">
```

<!ELEMENT Recipe (Name, Description?, Ingredients?, instructions?>)

The <!Element…> statement defines a tag in the document. This tag defines a <Recipe> tag, stating that it contains a <Name>, an optional <Description> (? Denotes optionality), and optional <Ingredients> and <Instructions> tags.

<!ELEMENT Name (#PCDATA)>

This simply states that a <Name> tag contains character data and nothing else.

<!ATTLIST Item optional CDATA "0" isVegetarian CDATA "true">

This section states that the <Item> tag has two possible attributes: optional, whose default value is 0, and isVegetarian, whose default value is true. These attribute values aren't limited to numeric values, they can be any text.

XML Schema

Better than DTD specification, especially if you come from the database world. "If DTD is a match, then XML-Schema is an atom bomb".

```
XML Fragment:
<InvoiceNo>123456</InvoiceNo>
<ProductID>BR549</ProductID>

DTD  <!ELEMENT InvoiceNo (#PCDATA)>
        <!ELEMENT ProductID (#PCDATA)>

XML Schema
<element name="InvoiceNo"
            type="positive-integer" />
<element name="ProductID"
            type="PCode"/>
<simpleType name="PCode" base="String">          <pattern value="[A-Z]{2}d{3}" />
</simpleType>
```

5) XML Parsers

- Available in most languages
- Browser and Server based parsers
- 2 Types
  - Non-Validating – Must be well formed
  - Validating – Must also comply with DTD/Schema
- 2 Standard API's
  - SAX – Simple API XML
  - DOM – Document Object Model

SAX – Simple API for XML

- Easiest way to get started
- Easy to use set of methods
  - startDocument()
  - startElement(name, AttributeList)
  - endElement(name)
- These methods are called as the parser reads each element
- Can handle any size of xml document

DOM – Document Object Model

- Entire document accessible in memory
- Document is tree of nodes
- Everything is a node, element name, text, even white space
- Fairly "Low level" api, 12 node types: Attribute Node, Cdata Node, Comment Node, Document Fragment Node, Document Node, Document Type Node, Element Node, Entity Node, Entity Reference Node, Notation Node, Processing Instruction Node, Text Node
- Easier to use XPATH to traverse document tree

Java and XML

- Many Java XML parsers available
- Oracle XDK – XML Developers Kit
  - Includes XML Parser
  - XSLT Transformation Engine
  - XSQL Database Data Publishing Framework
- Latest JDK (1.4) has xml support
- Also check out JDOM and JAXP

6) What happened to the "Presentation" of the data?

- GREAT SOLUTION!!!!
- XSLT – Transformation Language
- Language that allows you to present the data however you want to, extremely flexible, also handles restrictions and sorting
- Great for data conversion, you can "Transform" your xml document into ANY other text format, code, xml, html, comma separated, etc.
- Significant learning curve involved

Sample XSL (xpath highlighted)

```
<xsl:stylesheet version="1.0" … >
<xsl:template match="/">
<html><body><h2>Configuration Details</h2><table>
<td><xsl:value-of
        select="/page/request/parameters/server"/>:<xsl:value-of
        select="/page/request/parameters/port"/>
</td></tr>
<xsl:choose>
      <xsl:when test="/page/xsql-error/message[.='Connection refused']">
      <tr><td>The connection to the server was refused! </td></tr></xsl:when>
      <xsl:when test="starts-with(/page/xsql-error/message,'Error loading
    URL')"><tr><td><xsl:value-of select="/page/xsql-
    error/message"/></td></tr><tr><td>The server was unable to obtain the
    environment details.  The URL (above) may be incorrect or the server may not be
    configured properly.</td></tr> </xsl:when>
```

7) XML, Web Services & E-Commerce

- Web Services based on 4 standards
    - HTTP – Hyper Text Transfer Protocol
    - SOAP – Simple Object Access Protocol
    - UDDI – Universal Description, Discovery and Integration
    - WSDL – Web Service Description Language
- All of these except HTTP are XML based

HTTP/SOAP – Request Response

- Similar to current Web, request (URL) initiated from Browser, response returned by Web / Application Server
- Text based, runs on current web infrastructure, able to communicate through firewall

SOAP Request Example

```
POST /examples HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: 474
SOAPAction: "/examples"
```

```
<?xml version="1.0">
<SOAP-ENV:Envelope>
   <SOAP-ENV:Body>
       <getStateName>
                <statenum xsi:type="xsd:int">41</statenum>
        </getStateName>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Response Example

```
HTTP/1.1 500 Server Error
Content-Type: text/xml; charset=utf-8
Content-Length: 474

<?xml version="1.0">
<SOAP-ENV:Envelope>
   <SOAP-ENV:Body>
      <SOAP-ENV:Fault>
         <faultcode>SOAP-ENV:Client</faultcode>
         <faultstring>Can't call getStateName because there are too many parameters
         </faultstring>
      </SOAP-ENV:Fault>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

UDDI - Universal Description, Discovery and Integration

- Electronic XML Yellow Pages
- Describes the various interfaces to web services for a company
- Syntax XML based

WSDL - Web Services Description Language

- Describes the implementation of a web service, more detailed than UDDI

```
<?xml version="1.0"?>
<definitions name="CellPhoneService">
<portType name="CellPhoneService_port">
<operation name="getListOfModels">
<operation name="getPrice">
```

8) Summary

- XML great solution for many forms of data
- Isn't going to replace databases any time soon
- XSL makes the story even better, especially for Web Services/E-Commerce transformations
- Everyone in the IT industry will be / is being impacted by XML, EVERYONE
- HUGE number of areas we haven't covered, .NET, Namespaces, Docbook, XML Messaging, etc. etc. etc.

About the Author

Tony Obermeit, Senior Development Manager, Web ADI (Applications Desktop Integrator) Group, Oracle Corporation, Brisbane, Queensland, Australia

Tony has been employed as an I.T. professional in Brisbane for the past 20 years. For over 10 years now Tony has been working for Oracle Corporation in a variety of roles including Client Services Manager, Education Centre Manager and Applications Architect for the Web ADI product. In his role with Web ADI, Tony was involved the design and development of an XML based application using the Oracle XML Developers kit.