Introduction to XML and JSON

COURTESY FOR SOME OF THE SLIDES:

DR. CAROL WOLF

COMPUTER SCIENCE DEPARTMENT

PACE UNIVERSITY, NEW YORK, USA.

DR. LYNNE GREWE

DEPARTMENT OF COMPUTER SCIENCE

CAL STATE EAST BAY, HAYWARD, CA 94542.

What is XML

XML stands for eXtensible Markup Language.

A markup language is used to provide information about a document.

Tags are added to the document to provide the extra information.

HTML tags tell a browser how to display the document.

XML tags give a reader some idea what some of the data means.

What is XML Used For?

XML documents are used to transfer data from one place to another often over the Internet.

XML subsets are designed for particular applications.

One is RSS (Rich Site Summary or Really Simple Syndication). It is used to send breaking news bulletins from one web site to another.

A number of fields have their own subsets. These include chemistry, mathematics, and books publishing.

Most of these subsets are registered with the W3Consortium and are available for anyone's use.

Advantages of XML

XML is text (Unicode) based.

- Takes up less space.
- Can be transmitted efficiently.

One XML document can be displayed differently in different media.

- Html, video, CD, DVD
- You only have to change the XML document in order to change all the rest.

XML documents can be modularized. Parts can be reused.

Example of an HTML Document

```
<html>
 <head><title>Example</title></head.
<body>
 <h1>This is an example of a page.</h1>
 <h2>Some information goes here.</h2>
</body>
</html>
```

Example of an XML Document

```
<?xml version="1.0"/>
<address>
 <name>Alice Lee</name>
 <email>alee@aol.com</email>
 <phone>212-346-1234</phone>
 <birthday>1985-03-22</birthday>
</address>
```

Difference Between HTML and XML

HTML tags have a fixed meaning and browsers know what it is.

XML tags are different for different applications, and users know what they mean.

HTML tags are used for display.

XML tags are used to describe documents and data.

XML Rules

Tags are enclosed in angle brackets.

Tags come in pairs with start-tags and end-tags.

Tags must be properly nested.

- **<name><email>…</name></email> is not allowed.**
- **<name><email>…</email><name> is.**

Tags that do not have end-tags must be terminated by a ‘/’.

- <br /> is an html example.

More XML Rules

Tags are case sensitive.

- **<address> is not the same as <Address>**

XML in any combination of cases is not allowed as part of a tag.

Tags may not contain '<' or '&'.

Tags follow Java naming conventions, except that a single colon and other characters are allowed.

Tags must begin with a letter and may not contain white space.

Documents must have a single *root* tag that begins the document.

Encoding

XML (like Java) uses Unicode to encode characters.

Unicode comes in many flavors. The most common one used in the West is UTF-8.

UTF-8 is a variable length code. Characters are encoded in 1 byte, 2 bytes, or 4 bytes.

The first 128 characters in Unicode are ASCII.

In UTF-8, the numbers between 128 and 255 code for some of the more common characters used in western Europe, such as ã, á, å, or ç.

Well-Formed Documents

An XML document is said to be well-formed if it follows all the rules.

An XML parser is used to check that all the rules have been obeyed.

Recent browsers such as Internet Explorer 5 and Netscape 7 come with XML parsers.

Parsers are also available for free download over the Internet. One is Xerces, from the Apache open-source project.

Java 1.4 also supports an open-source parser.

XML Example Revisited

A flat text file is not so clear.

```
Alice Lee
alee@aol.com
212-346-1234
1985-03-22
```

The last line looks like a date, but what is it for?

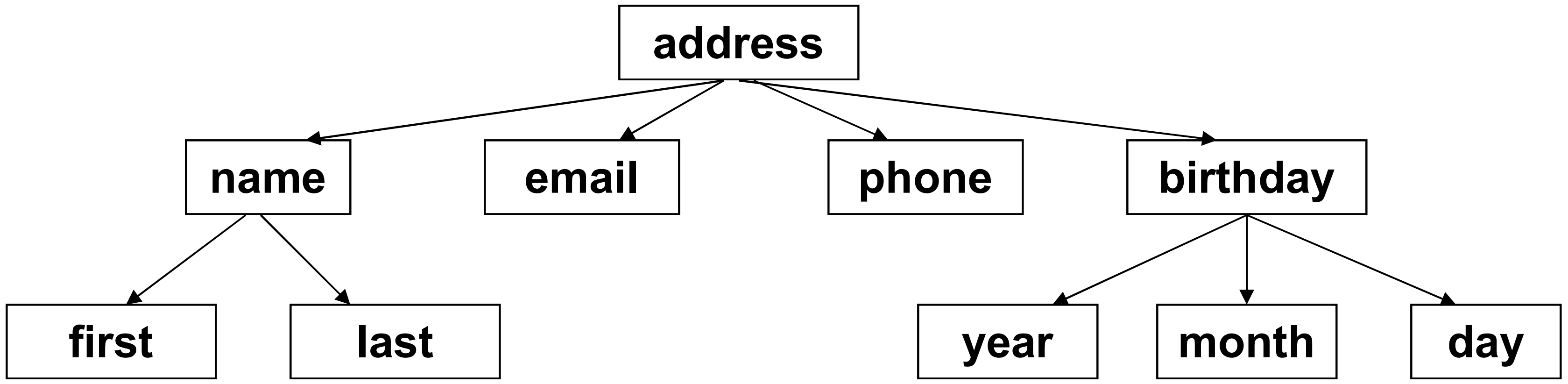
```
<?xml version="1.0"/>
<address>
  <name>Alice Lee</name>
  <email>alee@aol.com</email>
  <phone>212-346-1234</phone>
  <birthday>1985-03-22</birthday>
</address>
```

Markup for the data aids understanding of its purpose.

Expanded Example

```
<?xml version = “1.0” ?>
<address>
  <name>
      <first>Alice</first>
      <last>Lee</last>
    </name>
    <email>alee@aol.com</email>
    <phone>123-45-6789</phone>
    <birthday>
      <year>1983</year>
      <month>07</month>
      <day>15</day>
  </birthday>
</address>
```

XML Files are Trees



XML Trees

An XML document has a single root node.

The tree is a general ordered tree.

- A parent node may have any number of children.
- Child nodes are ordered, and may have siblings.

Preorder traversals are usually used for getting information out of the tree.

Validity

A well-formed document has a tree structure and obeys all the XML rules.

A particular application may add more rules in either a DTD (document type definition) or in a schema.

Many specialized DTDs and schemas have been created to describe particular areas.

These range from disseminating news bulletins (RSS) to chemical formulas.

DTDs were developed first, so they are not as comprehensive as schema.

Document Type Definitions

A DTD describes the tree structure of a document and something about its data.

There are two data types, PCDATA and CDATA.

- PCDATA is parsed character data – data to be parsed by XML parser.
- CDATA is character data, not usually parsed.

A DTD determines how many times a node may appear, and how child nodes are ordered.

DTD for address Example

```
<!ELEMENT address (name, email, phone, birthday)>
<!ELEMENT name (first, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT birthday (year, month, day)>
<!ELEMENT year (#PCDATA)>
<!ELEMENT month (#PCDATA)>
<!ELEMENT day (#PCDATA)>
```

Schemas

Schemas are themselves XML documents.

They were standardized after DTDs and provide more information about the document.

They have a number of data types including string, decimal, integer, boolean, date, and time.

They divide elements into simple and complex types.

They also determine the tree structure and how many children a node may have.

Schema for First address Example

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="address">
  <xs:complexType>
        <xs:sequence>
              <xs:element name="name" type="xs:string"/>
              <xs:element name="email" type="xs:string"/>
              <xs:element name="phone" type="xs:string"/>
              <xs:element name="birthday" type="xs:date"/>
        </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

Explanation of Example Schema

<?xml version="1.0" encoding="ISO-8859-1" ?>

ISO-8859-1, Latin-1, is the same as UTF-8 in the first 128 characters.

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

www.w3.org/2001/XMLSchema contains the schema standards.

<xs:element name="address">

<xs:complexType>

This states that address is a complex type element.

<xs:sequence>

This states that the following elements form a sequence and must come in the order shown.

<xs:element name="name" type="xs:string"/>

This says that the element, name, must be a string.

<xs:element name="birthday" type="xs:date"/>

This states that the element, birthday, is a date. Dates are always of the form yyyy-mm-dd.

XSLT

Extensible Stylesheet Language Transformations

XSLT is used to transform one xml document into another, often an html document.

The Transform classes are now part of Java 1.4.

A program is used that takes as input one xml document and produces as output another.

If the resulting document is in html, it can be viewed by a web browser.

This is a good way to display xml data.

A Style Sheet to Transform address.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:template match="address">
              <html><head><title>Address Book</title></head>
              <body>
                    <xsl:value-of select="name"/>
                    <br/><xsl:value-of select="email"/>
                    <br/><xsl:value-of select="phone"/>
                    <br/><xsl:value-of select="birthday"/>
              </body>
              </html>
        </xsl:template>
  </xsl:stylesheet>
```

The Result of the Transformation

Alice Lee
alee@aol.com
123-45-6789
1983-7-15

Parsers

There are two principal models for parsers.

SAX – Simple API for XML

- Uses a call-back method
- Similar to javax listeners

DOM – Document Object Model

- Creates a parse tree
- Requires a tree traversal

What is JSON

JavaScript Object Notation

Used to format data

Commonly used in Web as a vehicle to describe data being sent between systems

JSON example

"JSON" stands for "JavaScript Object Notation"

- Despite the name, JSON is a (mostly) language-independent way of specifying objects as name-value pairs

-
  ```
  {"skillz": {
      "web":[
          { "name": "html",
            "years": 5
          },
          { "name": "css",
            "years": 3
          }]
      "database":[
          { "name": "sql",
            "years": 7
          }]
  }}
  ```

JSON syntax

An *object* is an unordered set of name/value pairs

- The pairs are enclosed within braces, { }
- There is a colon between the name and the value
- Pairs are separated by commas
- Example: { "name": "html", "years": 5 }

An *array* is an ordered collection of values

- The values are enclosed within brackets, [ ]
- Values are separated by commas
- Example: ["html", ”xml", "css"]

JSON syntax

A *value* can be: A string, a number, true, false, null, an object, or an array

- Values can be nested

Strings are enclosed in double quotes, and can contain the usual assortment of escaped characters

Numbers have the usual C/C++/Java syntax, including exponential (E) notation

- All numbers are decimal--no octal or hexadecimal

Whitespace can be used between any pair of tokens

Comparison of JSON and XML

Similarities:

- Both are human readable
- Both have very simple syntax
- Both are hierarchical
- Both are language independent
- Both supported in APIs of many programming languages

Differences:

- Syntax is different
- JSON is less verbose
- JSON includes arrays
- Names in JSON must not be JavaScript reserved words
- XML can be validated

YAML – another option?

YAML can be taken as an acronym for either

- Yet Another Markup Language
- YAML Ain’t Markup Language

Like JSON, the purpose of YAML is to represent typical data types in human-readable notation

YAML is (almost) a superset of JSON, with many more capabilities (lists, casting, etc.)

- Except: YAML doesn’t handle escaped Unicode characters
- Consequently, JSON can be parsed by YAML parsers

When JSON isn’t enough, consider YAML

JSON and Java

HOW TO PARSE JSON IN JAVA

Mapping between JSON and Java entities

JSON	**Java**
string	java.lang.String
number	java.lang.Number
true|false	java.lang.Boolean
null	null
array	java.util.List
object	java.util.Map

XML and JSON Examples

See project XMLJSONParser

Javascript JSON examples:

https://www.w3schools.com/js/js_json_intro.asp

XML with JavaScript example:

http://www.peachpit.com/articles/article.aspx?p=29307&seqNum=4

References

Elliotte Rusty Harold, *Processing XML with Java,* Addison Wesley, 2002.

Elliotte Rusty Harold and Scott Means, *XML Programming*, O'Reilly & Associates, Inc., 2002.

W3Schools Online Web Tutorials, http://www.w3schools.com.