E06

XML 101

Benjamin Sheats

St. Louis, MO **Sept. 30 - Oct. 3, 2002**

RETURN TO INDEX

XML History

- SGML standardized by ISO in 1986
  - very complex and sophisticated
- HTML is an *application* of SGML
- XML is a simplified subset of SGML
  - stands for "Extensible Markup Language"
  - official W3C recommendation in 1998
  - more powerful than HTML, yet much simpler than SGML
  - XML is also an *application* of SGML

Interesting Factoid: MAGMA was the runner-up in the acronym race.
MAGMA stands for "Minimal Architecture for Generalized Markup Applications"

XML Syntax

- XML documents must be "well-formed"
  - only 1 top level element may be present (called the "root")
  - all start tags must have an equivalent end tag
  - empty tags can use a special syntax
  - attributes must be surrounded by single or double quotes

Sample XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Ben's Coffee Ratings -->
<ratings>
    <coffee name="Safeway Select Antigua">
        <blend>arabica</blend>
        <roast>medium</roast>
        <score unit="stars">3</score>
    </coffee>
</ratings>
```

Namespaces

- Required once XML documents are exchanged, or processed by more than one software package
- Analogous to Java packages, or C++ namespaces
- Defined using "URI's", or Uniform Resource Identifiers
  - in order to be equivalent, 2 URI's must match character for character (case sensitive)

Namespace Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Ben's Coffee Ratings -->
<ben:ratings xmlns:ben="http://www.sheats.net/coffee">
   <coffee name="Safeway Select Antigua"
           xmlns="http://www.sheats.net/coffee">
      <blend>arabica</blend>
      <roast>medium</roast>
      <score unit="stars">3</score>
   </coffee>
</ben:ratings>
```

XML Parsers - part 1

- 2 main types
  - DOM
    - most commonly used
    - parses entire document into a tree view in memory
    - can be inefficient if the document is very large
  - SAX
    - event based (callbacks)
    - useful for getting at specific elements, without loading the entire XML document

XML Parsers - part 2

- XML Parsers are widely available, for virtually all platforms
- Available in Java, C, C++, COBOL, etc...
- Built into modern web browsers
- Parsers are required to be strict
  - Smallest syntax error must result in an error
  - Contrast with HTML parsers, which are very fault-tolerant

XML Schemas and DTDs

- Used for XML validation
- Specifies the grammar and vocabulary
- Common in B2B scenarios
  - Companies can just agree on a Schema or DTD
- DTDs invented first (long before XML)
  - "Document Type Definition"
  - Part of XML 1.0 spec
  - Limited type system
  - Not XML compliant!
- Schemas are more flexible and powerful
  - XML compliant
  - Robust type system

Sample DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ratings [
    <!ELEMENT ratings (coffee)*>
    <!ELEMENT coffee (blend, roast, score)>
    <!ELEMENT blend (#PCDATA)>
    <!ELEMENT roast (#PCDATA)>
    <!ELEMENT score (#PCDATA)>
    <!ATTLIST coffee
        name CDATA #REQUIRED>
    <!ATTLIST score
        units CDATA "stars">
]>
<!-- Ben's Coffee Ratings -->
<ratings>
    ....
</ratings>
```

Sample XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.sheats.net/coffee"
            xmlns="http://www.sheats.net/coffee"
            elementFormDefault="qualified">
  <xsd:element name="ratings">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="coffee" type="coffeeType">
          <xsd:attribute name="name" type="xsd:string"
                         use="required"/>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

```
<xsd:complexType name="coffeeType">
  <xsd:sequence>
    <xsd:element name="blend" type="xsd:string"/>
    <xsd:element name="roast" type="roastType"/>
    <xsd:element name="score" type="scoreType"/>
  </xsd:sequence>
</xsd:complexType>
```

```
<xsd:simpleType name="roastType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="light"/>
    <xsd:enumeration value="medium"/>
    <xsd:enumeration value="dark"/>
  </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:complexType name="scoreType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:integer">
      <xsd:attribute name="units" type="xsd:string"
                     default="stars"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

Web Services

- Modular, self-describing applications
  - Defined with WSDL files
    - "Web Services Description Language"
- Can be published, located, and invoked
  - UDDI registry
    - Universal Discovery Description and Integration
- Most common use is SOAP over HTTP
  - However any RPC and transport protocol can be used

SOAP

- "Simple Object Access Protocol"
- Really just a means of doing a remote procedure call
- Invented before Web Services
  - however, WSDL is so far the best metadata implementation for SOAP
- Uses XML Schema type system
  - 2 additions
    - arrays
    - typed references

Sample WSDL - part 1

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="CoffeeRatings" targetNamespace="http://www.sheats.net/coffee"
             xmlns:tns="http://www.sheats.net/coffee"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:xsd="http://www.w3.org/2001/XMLSchema"
             xmlns="http://schemas.xmlsoap.org/wsdl/">
  <message name="getCoffeeRatingsRequest">
    <part name="name" type="xsd:string"/>
  </message>
  <message name="getCoffeeRatingsResponse">
    <part name="score" type="xsd:integer"/>
  </message>
  <portType name="CoffeeRatingsPortType">
    <operation name="getCoffeeRatings">
      <input message="tns:getCoffeeRatingsRequest"/>
      <output message="tns:getCoffeeRatingsResponse"/>
    </operation>
  </portType>
```

Sample WSDL - part 2

```
<binding name="CoffeeRatingsBinding" type="CoffeeRatingsPortType">
  <soap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="getCoffeeRatings">
    <soap:operation soapAction="urn:coffeeRatings"/>
    <input>
      <soap:body use="encoded" namespace="urn:coffeeRatings"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </input>
    <output>
      <soap:body use="encoded" namespace="urn:coffeeRatings"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
    </output>
  </operation>
</binding>
<service name="CoffeeRatingsService">
  <port name="CoffeeRatingsPort" binding="CoffeeRatingsBinding">
    <soap:address location="http://www.sheats.net:9090/soap"/>
  </port>
</service>
</definitions>
```

XHTML

- Future of HTML
- Defines organization, not presentation
- Fully XML compliant
  - radically reduces browser complexity
  - much simpler for non-browser agents to parse and understand
- All elements are now defined as lowercase
  -
 is no longer valid

Interesting Note:
 is somewhat incompatible with today's browsers. However,
 (with space) is, and is also compliant with XML parsers!

Sample XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
    <head>
        <title>Ben's Coffee Ratings</title>
    </head>
    <body>
        <p>Safeway Select Antigua
            <ul>
                <li>Blend = arabica</li>
                <li>Roast = medium</li>
                <li>Score = 3 stars</li>
            </ul>
        </p>
    </body>
</html>
```

CSS

- "Cascading Style Sheets"
- Intended to be used in conjunction with HTML (XHTML) files to add presentation information
- Normally stored in an external file
  - this allows look-and-feel to easily extend to a large range of HTML (XHTML) files
- Called "cascading" because style is inherited
  - Browser default
  - External style sheet
  - Internal (within the html) style sheet
  - Individual element style

Sample CSS

```
<style type="text/css">
    body {background-color: yellow}
    p {text-decoration: underline}
    li {font-style: italic}
</style>
```

Safeway Select Antigua

- *Blend = arabica*
- *Roast = medium*
- *Score = 3 stars*

XSL

- "eXtensible Stylesheet Language"
- Consists of 3 parts
  - XSLT
  - XPath
  - XSL Formatting Objects
- XSLT is used for transforming XML documents
- XSL Formatting Objects are used for displaying XML (compare to HTML and CSS)

XSLT

- Transforms a source XML document into another document
  - Most common usage is XML to XHTML
- Can add, delete, rearrange, and sort elements
- Uses XPath to locate elements with the source XML document

XPath

- Syntax for defining parts of an XML document
- Uses "paths", much like file system paths, for locating elements
- Defines a library of functions to enhance path logic
- Complicated and sophisticated

Sample XSLT + XPath

```
<?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="/">
      <html>
         <head><title>Ben's Coffee Ratings</title></head>
         <body>
            <xsl:for-each select="ratings/coffee">
               <p><xsl:value-of select="@name"/>
                  <ul>
                     <li>Blend = <xsl:value-of select="blend"/></li>
                     <li>Roast = <xsl:value-of select="roast"/></li>
                     <li>Score = <xsl:value-of select="score"/>
                                 <xsl:value-of select="score/@units"/>
                     </li>
                  </ul>
               </p>
            </xsl:for-each>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

XLink

- Used to provide links between XML documents
- Like <a href> in html
- Much more powerful and complicated
  - links can have multiple sources, destinations, etc...
- Not backwards compatible with href

Sample XLink

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Ben's Coffee Ratings -->
<ratings xmlns="http://www.sheats.net/coffee"
          xmlns:xlink = "http://www.w3.org/1999/xlink">
   <coffee name="Safeway Select Antigua"
          xlink:type="simple"
          xlink:href="http://www.safeway.com">
      <blend>arabica</blend>
      <roast>medium</roast>
      <score unit="stars">3</score>
   </coffee>
</ratings>
```

XPointer

- Extension of XPath
- Used by XLink to point to external XML document fragments
- An XPath expression with a URI in front of it

```
<coffee name="Safeway Select Antigua"
        xlink:type="simple"
        xlink:href="coffee.xml#xpointer(//safeway/select/antigua)">
</coffee>
```

URI

XPath expression

XMI

- "XML Metadata Interchange"
- Intended for exchange of metadata between modeling tools
- Merging of 3 key technologies
  - XML - Extensible Markup Language
  - UML - Unified Modeling Language
  - MOF - Meta Object Facility
- CAM initiative at IBM
  - Language metadata models used for data transformation

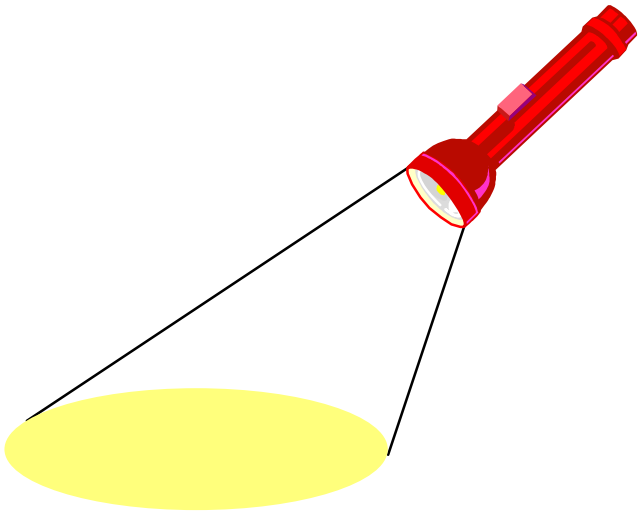
Sample XMI

```
<?xml version="1.0" encoding="UTF-8"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI" xmlns:MFS="MFS.xmi">
   <MFS:MFSMessage xmi:id="MFSMessage_1" label="IVTNO" type="output">
      <logicalPages xmi:id="MFSLogicalPage_1">
         <segments xmi:id="MFSSegment_1">
            <messageFields xmi:id="MFSMessageField_1" length="40"/>
            <messageFields xmi:id="MFSMessageField_2" length="8"/>
            <messageFields xmi:id="MFSMessageField_3" length="10"/>
            <messageFields xmi:id="MFSMessageField_4" length="10"/>
            <messageFields xmi:id="MFSMessageField_5" length="10"/>
            <messageFields xmi:id="MFSMessageField_6" length="7"/>
            <messageFields xmi:id="MFSMessageField_7" length="4"/>
            <messageFields xmi:id="MFSMessageField_8" systemLiteral="date2"/>
         </segments>
      </logicalPages>
      <nextMessage href="IVTNOMI1.xmi#MFSMessage_1"/>
   </MFS:MFSMessage>
</xmi:XMI>
```

XML and IMS

- WebSphere family
  - WebSphere Application Developer - Integration Edition
    - Publishing existing COBOL and MFS applications as Web Services
  - WebSphere Application Server
- XML Parser in COBOL

Questions?



RETURN TO INDEX