Users’ Conference
Jerusalem, Israel
September 2000
XML Tutorial

FAME Users’ Conference
Jerusalem, Israel
September 2000
**What is XML?**

- stands for Exensible Markup Language
- is a *meta-language* for creating and formatting document markups
- is a simplified form of SGML – Standard Generalized Markup Language
- is a system to unambiguously identify data components within a document
- XML and related standards are administered by the World Wide Web Consortium (W3C – www.w3c.org) - the keeper of HTML
Why all the excitement?

- The promise of universal access to data
- Convergence of concepts
  - Distributed data
    - It is no longer feasible to think of all the data in a single container or all residing at the user’s site
  - Distributed computation
  - Open standards
  - The Internet
- This means you can share data no matter what database technologies, programming languages or platforms you use, provided you can come to some agreement on naming conventions and categorization of data items.
What is the reality?

• It is still early in the game
• *But* there is already huge interest by commercial, non-profit and governmental organizations
• Major complaint
  – XML is verbose
    • but there are strategies to deal with that
      – on the fly compression techniques
<?xml version="1.0" encoding="UTF-8"?>
<TimeSeries id="israel.interestrate" frequency="MONTHLY" type="FLOAT">
    <description>Israel Central Bank Interest Rates</description>
    <observations>
        <observation>
            <date>2000-01</date> <value>10.7</value> <status>OK</status>
        </observation>
        <observation>
            <date>2000-02</date> <value>10.3</value> <status>OK</status>
        </observation>
        <observation>
            <date>2000-03</date> <value>9.9</value> <status>OK</status>
        </observation>
    </observations>
</TimeSeries>
XML Terminology

• An XML document consists of one or more elements

• An element consists of:
  – *Opening tag* – “<“ name of tag “>”
  – *Text*
  – *Closing tag* – “</” name of tag “>”

• Elements can have *attributes* applied

<description lang="en">
Israel Central Bank Interest Rates
</description>
### XML - elements

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Israel Central Bank Interest Rates</td>
</tr>
</tbody>
</table>

- **Simple element**
  - contains just string data

```
<observations>
  <observation>
    <date>2000-01</date>
    <value>10.7</value>
    <status>OK</status>
  </observation>
</observations>
```

- **Complex element**
  - contains other complex and simple elements
  - must resolve to a simple element
XML - elements

<logo src="logo.gif"/>
  or
<logo src="logo.gif"> </logo>

• Empty element
  – contains just attributes – note different forms
XML - attributes

```xml
<TimeSeries id="israel.interestrate"
    frequency="MONTHLY"
    type="FLOAT">
...
</TimeSeries>
```

- restricted to strings
- typically descriptive of the container
XML - declaration

- Special information for the XML processor
- Format
  - `<?xml version="number"` 
    `[encoding="encoding"]` 
    `[standalone="yes|no"]` `?>`
- Version - Required, 1.0 is the current version
- Encoding - Character set used in file
  - Alternatives
    - UTF-8, UTF-16, ISO8859-1, ISO8859-8, US-ASCII, ...
- Standalone - Indicates whether a DTD is required
- Example

```xml
    <?xml version="1.0" encoding="UTF-8"?>
```
XML Rules

- Attributes must be in quotation marks
  - Single or double quotes allowed
- A non-empty element must have an opening and closing tag
- An empty element must have a slash (/) before the closing tag
- Tags must be nested correctly
  - `<Italic><Bold>some text</Bold></Italic>
  - Not
  - `<Italic><Bold>some text</Italic></Bold>
- Isolated markup characters (<, &) not allowed in text
- Only one root element
<?xml version="1.0" encoding="UTF-8"?>
<TimeSeries id="israel.interestrate" frequency="MONTHLY" type="FLOAT">
  <description>Israel Central Bank Interest Rates</description>
  <observations>
    <observation>
      <date>2000-01</date> <value>10.7</value> <status>OK</status>
    </observation>
    <observation>
      <date>2000-02</date> <value>10.3</value> <status>OK</status>
    </observation>
    <observation>
      <date>2000-03</date> <value>9.9</value> <status>OK</status>
    </observation>
  </observations>
</TimeSeries>
XML – instructions (optional)

- `<?xml ... ?>`
  - NOT OPTIONAL
- `<!DOCTYPE ... >`
  - used to reference DTD/schema
- `<!-- ... -->`
  - comments
- `<? ... ?>`
  - processing instructions for external programs (e.g. MS Word)
- `<![CDATA[ ... ]]>`
  - Special sections of character data which may have reserved markup symbols
XML - Namespaces

- Provides the ability to use multiple XML vocabularies
- Resolves identical element names that represent conceptually distinct entities
- Uses URIs – Universal Resource Identifiers
  - URLs – Universal Resource Locators
    - “http://” <host> [“:” <port>] [<path> [“?” <query>]]
  - URNs – Universal Resource Names
    - “urn:” <NID> ”:” <NSS>
      - Name Space Idenifier : Name Space Specific String
      - Microsoft based example – the COM interface iUnknown
        - urn:uuid:00000000-0000-0000-C000-0000000000046
XML - Namespaces

• Example
  – Reference the status entities defined by FAME
  – “OK”, “NA”, “NC”, “ND”, “NIC”

```xml
<?xml version="1.0" encoding="UTF-8"?>
<TimeSeries id="israel.interestrate" frequency="MONTHLY" type="FLOAT"
    xmlns:fs="http://www.fame.com/xml/statuses">
  ...
  <observations>
    <observation>
      <date>2000-01</date>
      <value>10.7</value>
      <fs:status>OK</fs:status>
    </observation>
    ...
  </observations>
</TimeSeries>
```
XML so far

• Fairly simple and straightforward
• A very flexible structure
• A minimal number of keywords and acronyms

• Unfortunately that is just the beginning
• What XML has become is
An Alphabet Soup

If you like acronyms, you’ll love XML!

- XML
- DTD
- URIs, URNs, URLs
- XSL
- XSLT
- XML Schema
- XLink
- Xpointer
- BizTalk Schema
- DOM
- SAX
- CDF
- RDF
- Xforms
- MathML
- XMI
- XML Signature
- SOAP

And more on the way
XML with some order

- XML — document itself
- Namespaces — reference other XML models
- Validation — validate formatting of XML document
- Processing APIs — computer models DOM and SAX
- Transformation — convert to other XML or HTML
- Linking — pointers to other documents of parts thereof
Other tools

- XML Editors/Viewers
- XML Applications – RDF, XMI
- Internet - XHTML
- Object Protocols - SOAP
- Document definition repositories
  - Key activity defining naming conventions, i.e. DTD/Schema
  - OASIS, W3C, FixML, …
XML - Validation

- Documents that specify the proper form for specific XML documents
- DTD
  - Document Type Definition
- XML Data
  - Microsoft’s first pass at XML Schema
- BizTalk Schema
- XML Schema
  - In standards organization – www.w3c.org
Files have "*.dtd" as standard suffix

DTD commands are not XML
  Therefore 2 parsers are required to validate

Cannot specify details of element data, must be done in program
XML Schema

- Files have “.xsd” as standard suffix
- XML Schema is XML compliant
- XML’s next generation of validation standard
- XML Schema standard administered by World Wide Web Consortium
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE xsd:schema PUBLIC "-//W3C//DTD XMLSCHEMA 19991216//EN" "">

<!ENTITY % p 'xsd:'>
<!ENTITY % s ':xsd'>

<xsd:schema xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <xsd:element name="TimeSeries">
    <xsd:complexType content="elementOnly">
      <xsd:element name="description" type="xsd:string"/>
      <xsd:element name="observations" type="observationsType"/>
      <xsd:attribute name="id" type="xsd:string" use="required"/>
      <xsd:attribute name="frequency">
        <xsd:simpleType base="xsd:NMTOKEN">
          <xsd:enumeration value="MONTHLY"/>
          <xsd:enumeration value="QUARTERLY"/>
          <xsd:enumeration value="ANNUAL"/>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
<xsd:complexType name="observationType" content="elementOnly">
    <xsd:sequence>
        <xsd:element name="date" type="xsd:date"/>
        <xsd:element name="value" type="xsd:string"/>
        <xsd:element name="status">
            <xsd:simpleType base="xsd:string">
                <xsd:enumeration value="OK"/>
                <xsd:enumeration value="NA"/>
                <xsd:enumeration value="NC"/>
                <xsd:enumeration value="ND"/>
                <xsd:enumeration value="NIC"/>
            </xsd:simpleType>
        </xsd:element>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="observationsType" content="elementOnly">
    <xsd:element name="observation" type="observationType" minOccurs="1" maxOccurs="unbounded"/>
</xsd:complexType>
</xsd:schema>
XSL – Extensible Stylesheet Language

- Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3c.org/TR/WD-xsl">
  <xsl:template match="/">
    <html>
      <body>
        <table border="2" bgcolor="yellow">
          <tr>
            <th>Date</th> <th>Value</th> <th>Status</th>
          </tr>
          <xsl:for-each select="TimeSeries/observations/observation">
            <tr>
              <td><xsl:value-of select="date"></td>
              <td><xsl:value-of select="value"></td>
              <td><xsl:value-of select="status"></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```
XSLT - XSL Transformations

- Extension of XSL
- Used to convert one XML document into another
- Used for more sophisticated transformations to HTML, such as table manipulations
APIs – SAX and DOM

• SAX – Simple API for XML
  – Event driven model
  – Callback occurs for each markup event
    • startDocument
    • endDocument
    • startElement
    • endElement
    • attribute
    • charData
    • ...
  – Developer must implement state-machine to handle input stream
  – Good for handling very large XML documents
    • E.g. an XML based data feed
APIs – SAX and DOM

- DOM – Document Object Model
  - Object model
  - Entire XML document is read and an object tree is created
  - Developer has random access to the documents contents
  - Easier to program than SAX but restricted to small documents
APIs – SAX and DOM

• DOM – Document Object Model
  – Multiple levels have been defined or are in the process of definition
    • DOM1
      – Core document models XML and HTML
    • DOM2
      – Adds filters, event model, supports namespaces
    • DOM3
      – Document loading and saving, validation, formatting
    • DOM4
      – User interface, prompting, multithreading, security
SOAP

- Simple Object Access Protocol
  - Used to invoke methods of objects on a server and return the result
- An alternative to CORBA and COM+
- Language and operating system neutral
- RPC over HTTP
  - HTTP header contains server identification, request type and status
  - SOAP Envelope contains namespace and encoding information
  - SOAP Body contains request
    - Request - Object ID, Method ID, input parameters
    - Response - Result

```
HTTP Header
 <SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
   ...
  </SOAP-ENV:Body>
 </SOAP-ENV:Envelope>
```
What is FAME going to do with XML?

- Define DTD’s and schema for FAME objects
- Provide Java components that render FAME objects as XML
- Add support to DataTone server for SOAP
  - Waiting for specification to stabilize
  - SOAP provides another API interface into Microsoft development languages (VB & C++)