XML for Java Developers  
G22.3033-002  

Session 5 - Main Theme  
XML Information Processing (Part I)  

Dr. Jean-Claude Franchitti  

New York University  
Computer Science Department  
Courant Institute of Mathematical Sciences

---

**Agenda**

- Summary of Previous Session
- XML applications development tools for Java
- XML application Development using the XML Java APIs
- Java-based XML application support frameworks

---

**Summary of Previous Session**

- XML-Based Software Development
- Business Engineering Methodology
- XML Metadata Management
- XML Linking/Pointer Language
- XML Data Binding
- Industry Specific Markup Languages
- Assignment 2a+2b (due next week)
XML-Based Software Development

- Business Engineering Methodology
  - Language + Process + Tools
  - e.g., Rational Unified Process (RUP)
- XML Application Development Infrastructure
  - Metadata Management (e.g., XMI)
  - XML APIs (e.g., JAXP, JAXB)
  - XML Tools (e.g., XML Editors, XML Parsers)
- XML Applications:
  - Application(s) of XML
  - XML-based applications/services (markup language mediators)
    - MOM & POP
    - Other Services (e.g., persistence, transaction, etc.)
  - Application Infrastructure Frameworks

More on XML Information Modeling

- Using UML use cases to support the development of DTDs and XML Schemas
- Establish linking relationship
  - See Family tree application of XML

Part I

XML Application Development Tools for Java
Java-enabled XML Technologies

- XML provides a universal syntax for Java semantics (behavior)
  - Portable, reusable data descriptions in XML
  - Portable Java code that makes the data behave in various ways
- XML standard extension
  - Basic plumbing that translates XML into Java
    - parser, namespace support in the parser, simple API for XML (SAX), and document object model (DOM)
  - XML data binding standard extension

XML Processors Characteristics

- An XML engine is a general purpose XML data processor
- An XML processor/parser is a software engine that checks the syntax (well-formedness) of XML documents
- If a schema (or DTD) is included, the parser can optionally validate the correctness of XML documents’ structure against it
- A parser reads the XML document’s information and makes it accessible to the XML application via a standard API

Common XML APIs

- Document Object Model (DOM) API
  - Tree structure-based API
  - Issued as a W3C recommendation (10/98)
  - See Session 5 Sub-Topic 1 Presentation
- Simple API for XML (SAX)
  - Event-driven API
  - Developed by David Meginson
  - ElementHandler API
    - Event-driven proprietary API provided by IBM’s XML4J
- Pure Java APIs: JDOM (Open Source) and JAXP
Java API Packages

- **java.xml.parsers**
  - The JAXP APIs, which provide a common interface for different vendors' SAX and DOM parsers.
  - Two vendor-neutral factory classes: `SAXParserFactory` and `DocumentBuilderFactory`, that give you a SAXParser and a DocumentBuilder, respectively. The `DocumentBuilderFactory` in turn, creates DOM-compliant `Document` object.
- **org.w3c.dom**
  - Defines the Document class (a DOM), as well as classes for all of the components of a DOM.
- **org.xml.sax**
  - Defines the basic SAX APIs.
- **jaxax.xml.transform**
  - Defines the XSLT APIs that let you transform XML into other forms.

Simple API for XML (SAX) Parsing APIs

SAX API Packages

- **org.xml.sax**
  - Defines the SAX interfaces.
- **org.xml.sax.ext**
  - Defines SAX extensions that are used when doing more sophisticated SAX processing, for example, to process a document type definitions (DTD) or to see the detailed syntax for a file.
- **org.xml.sax.helpers**
  - Contains helper classes that make it easier to use SAX – for example, by defining a default handler that has null-methods for all of the interfaces, so you only need to override the ones you actually want to implement.
- **javax.xml.parsers**
  - Defines the SAXParserFactory class which returns the SAXParser. Also defines exception classes for reporting errors.
DOM Parsing APIs

- `org.w3c.dom` - Defines the DOM programming interfaces for XML (and, optionally, HTML) documents, as specified by the W3C.
- `javax.xml.parsers` - Defines the `DocumentBuilderFactory` class and the `DocumentBuilder` class, which returns an object that implements the W3C `Document` interface. The factory that is used to create the builder is determined by the `javax.xml.parsers` system property, which can be set from the command line or overridden when invoking the `newInstance` method. This package also defines the `ParserConfigurationException` class for reporting errors.

XSLT APIs

- `TransformerFactory` - A factory for creating `Transformer`s.
- `Source` - Represents the source of the XSLT transformation.
- `Transformer` - The object that performs the actual transformation.
- `Result` - Represents the result of the transformation.
XSLT API Packages

- **javax.xml.transform**
  - Defines the TransformerFactory and Transformer classes, which you use to get a object capable of doing transformations. After creating a transformer object, you invoke its transform() method, providing it with an input (source) and output (result).
- **javax.xml.transform.dom**
  - Classes to create input (source) and output (result) objects from a DOM.
- **javax.xml.transform.sax**
  - Classes to create input (source) from a SAX parser and output (result) objects from a SAX event handler.
- **javax.xml.transform.stream**
  - Classes to create input (source) and output (result) objects from an I/O stream.

JAXP and Associated XML APIs

- **JAXP**: Java API for XML Parsing
  - Common interface to SAX, DOM, and XSLT APIs in Java, regardless of which vendor's implementation is actually being used.
- **JAXB**: Java Architecture for XML Binding
  - Mechanism for writing out Java objects as XML (marshalling) and for creating Java objects from such structures (unmarshalling).
- **JDOM**: Java DOM
  - Provides an object tree which is easier to use than a DOM tree, and it can be created from an XML structure without a compilation step.
- **JAXM**: Java API for XML Messaging
  - Mechanism for exchanging XML messages between applications.
- **JAXR**: Java API for XML Registries
  - Mechanism for publishing available services in an external registry, and for consulting the registry to find those services.

Content of Jar Files

- **jaxp.jar** (interfaces)
  - java.xml.parsers
  - java.xml.transform.dom
  - java.xml.transform.sax
  - java.xml.transform.stream
- **crimson.jar** (interfaces and helper classes)
  - org.xml.sax
  - org.xml.sax.helpers
  - org.xml.sax.ext
  - org.w3c.dom
- **xalan.jar** (contains all of the above implementation classes)
Sample XML parsers and engines

- XML parsers
  - RXP, Dan Connolly’s XML parser, XML-Toolkit, LXML, expat, TCLXML, xparse, XP, DataChannel, XPLparser (DXP), XML.Parse, PyXMLTok, Lark, Microsoft’s XML parser, IBM’s XML for Java, Apache’s Xerces-J, Arefed, xmlproc, xmllib, Windows foundation classes, Java Project X Parser (Crimson), OpenXML Parser, Oracle XML Parser, etc.
- SGML/XML parsers
  - SGMLSpn, SP

Sample XML Parsers and Engines (continued)

- XML middleware: Xpublish (Media Design), XML middleware 1.0
- DSSSL engines: Jade 1.1, DAE SDK, DAE Server SDK
- XSL processors: Sparse, Microsoft XSL processor, doproc, xslj, LotusXSL, Xalan, XSL-P
- XLink processors: xmlinks

Comprehensive List of XML Processors

- A comprehensive list of parsers is available at [http://www.xmlsoftware.com/parsers](http://www.xmlsoftware.com/parsers)
- Includes links to latest product pages
- Includes Version numbers, Licensing information, and Platform details
- Research work being done around MetaParsers and parallel XML parsers
Mainstream Java-Based XML Processors

- Sun’s Java Project X-Parser
  - Donated on April 13, 2000 to the Apache’s XML Project under the name “Crimson”
- Apache’s XercesJ
  - XercesJ is strongly recommended for this course
- Oracle’s XML Parser for Java
- Expat

Other Java-Based XML Processors

- Sun’s JAXP
- Jason Hunter and Brett McLaughlin’s OpenSource JDOM
- IBM Alphaworks’s XML for Java (XML4J)
  - Based on the Apache Xerces XML Parser
- DataChannel’s XJParser

XML Data Binding Standard Extension

- Aims to automatically generate substantial portions of the Java platform code that processes XML data
- A Sun project, codenamed “Adelard”
- See JSR-31 XML Data Binding Specification
- See http://java.sun.com/xml/jaxp-1.0/docs/binding/DataBinding.html
Part II

XML Application Development
Using the XML Java APIs

Typical XML Processor Installation

- Pick a processor based on the features it provides to match your requirements
- Download and install the latest (or supported) version of the JDK from http://www.javasoft.com
- Install the XML processor
- Update the PATH and CLASSPATH variables as needed, and test the processor

Reading XML Documents

- Use Apache’s XercesJ or Alphaworks’ XML Parser for Java
  - The “SimpleParseJava” application provided in section 2.4 of “XML and Java” will need to be adapted to support the latest version of the parsers
  - We suggest looking at the source for the sample applications located in XercesSamples.jar
  - For testing, use XML and Java’s sample document or the “personal.xml” sample XML document provided with XML4J
Presenting XML Documents Using Java Tools

- Presenting an XML document requires processing of the XML document by accessing its internal structure
- An XML document’s structure can be accessed using the various XML APIs
- Various third party tools have been implemented using such APIs to apply XSL style sheets to XML documents and generate HTML output (e.g., Xalan, LotusXSL)

XML Data Exchange Protocols

- Message format alternatives
  - Text-based (e.g., EDI, RFC822, SGML, XML)
  - Binary (e.g., ASN.1, CORBA/IIOP)
- See XML and Java sections 7.2, and 7.4
- An API that provide a common interface to work with EDI or XML/EDI objects is supported by OpenBusinessObjects
- Guidelines for using XML for EDI are provided at http://www.geocities.com/WallStreet/Floor/5815/guide.htm and http://www.xmledi-group.org/

XML Fragment Interchange

- Defines a way to send fragments of an XML document without having to send all of the containing document up to the fragment
- Fragments are not limited to predetermined entities
- The approach captures the context that the fragment had in the larger document to make it available to the recipient
- See http://www.w3.org/TR/WD-xml-fragment
XML APIs Characteristics

- **DOM API**: (See [http://www.developerlife.com/domintro/default.html](http://www.developerlife.com/domintro/default.html))
  - In DOM, an XML document is represented as a tree, which becomes accessible via the API.
  - The XML processor generates the whole tree in memory and hands it to an application program.

- **SAX API**: (See [http://java.sun.com/xml/docs/tutorial/sax/index.html](http://java.sun.com/xml/docs/tutorial/sax/index.html))
  - Does not generate a data structure.
  - Scans an XML document and generate events as elements are processed.
  - Events can be trapped by an application program via the API.

- **ElementHandler**
  - Event-driven like SAX, but also creates a DOM tree.
  - Open Source Pure Java API (JDOM)

Related Java Bindings

- Sun’s Java API for XML Parsing (JAXP)
  - Provides a standard way to seamlessly integrate any XML-compliant parser with a Java application.
  - Developers can swap between XML parsers without changing the application.
  - The reference implementation uses Sun’s Java Project X as its default XML parser.

- **DOM 2.0 and DOM 1.0 Java binding specification** ([http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/java-binding.zip](http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/java-binding.zip))

XML Data Processing Examples

- Section 2.7 of “XML and Java” covers various examples of XML document processing using the DOM, SAX, and ElementHandler APIs.

- Session 2’s Sub-Topic 2.2.8.1 on “Enterprise Application Integration with XML and Java” illustrates the use of XML for data interchange.
Part III

Java-Based Application Support Frameworks

XML MOM and POP Frameworks

- An XML support framework must include:
  - XML Parser (conformity checker)
  - XML applications that use the output of the Parser to achieve unique objectives
  - See sub-section 2.3.2 of the weekly notes on “XML MOM Application Server Frameworks” for a complete description of a general purpose XML MOM framework

POP Applications Support Frameworks

- Objective is to “serve” XML
- HTML generation applications are provided
- Sample solutions
  - XML::Parser module with Perl
  - XML processing via Java servlets
    - e.g., IBM Alphaworks’ XMLEnabler
    - See session 2’s sub-topic 2.3.2 on “XML POP Application Server Framework”
  - Apache’s Cocoon
  - Active Server Pages (ASP) with MSXML (see “Serving XML with ASP”, and rocket
MOM Applications Support Frameworks

- Many applications can be envisioned
  - One objective is to support application integration via XML data interchange
- Sample solutions:
  - XML::Parser module with Perl
  - XML processing via Java applications

Part IV
Conclusions

Summary

- Mainstream MOM and POP application development tools are being supported by IBM, Sun, Oracle, and Microsoft
- Java MOM and POP applications are developed using Java bindings to the DOM, and SAX APIs
- XML provides a standard data interchange message format
Summary (continued)

- The W3C XML-Fragments specification focuses on the handling of XML document fragments.
- MOM and POP (Java-based) application support frameworks are still emerging and are becoming common facilities in the ubiquitous Web Services Infrastructure.

More on Industry-Specific Markup Languages

(see http://www.oasis-open.org/cover/xml.html#contentsApps)

- Extensible Business Reporting Language (XBRL)
- Bank Internet Payment System (BIPS)
- Electronic Business XML (EbXML)
- Privacy-Enabled Customer Data Interchange (CPExchange)
- Visa XML Invoice Specification
- Legal XML
- NewsML
- Electronic Catalog XML (eCX)
- Open eBook Publication Structure

Sample XML-Based Architecture
Readings

- XML Development with Java: Chapters 2, 3, and 6-7
- Professional Java XML: Chapters 3, 4, and Appendices B, D, and F
- XML and Java: Chapter 2, 3, 4, and Appendices B-F
- Handouts posted on the course web site

Project Frameworks Setup (ongoing)

- Apache’s Web Server, TomCat/JRun, and Cocoon
- Apache’s Xerces, Xalan, Saxon
- Antenna House XML Formatter, Apache’s FOP, X-smiles
- Visibroker 4.5, WebLogic 6.1
- POSE & KVM (See Session 3 handout)

Assignment

Assignment #3:
- This part of the project focuses on the application process model development using XML information processing technology. The design development process should adhere to the following steps: (a) Identifying the points of data integration, (b) Defining the optimal integration approach at each point, (c) Establishing linking relationships, and (d) Considering data integration and linking issues when designing an overall application data model
- More specific project related information, and extra credit assignments will be provided during the session

Next Session:
XML Information Processing (Part II)

- Advanced XML Parser Technology
  - JDOM: Java-Centric API for XML
  - JAXP: Java API for XML Processing
  - DOM, SAX, JDOM, and JAXP comparison
- Latest W3C APIs and Standards for Processing XML
  - XML Infoset, DOM Level 3, Canonical XML
  - XML Signatures, XBase, XInclude, Xpointers
  - XML Fragments, XML Schema Adjuncts