XML for Java Developers
G22.3033-002

Session 6 - Main Theme
XML Information Processing (Part II)

Dr. Jean-Claude Franchitti

New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Agenda

- Summary of Previous Session
- Document Object Model (DOM)
- Advanced XML Parser Technology
  - JDOM: Java-Centric API for XML
  - JAXP: Java API for XML Processing
  - Parsers comparison
- Latest W3C APIs and Standards for Processing XML
  - XML Infoset, DOM Level 3, Canonical XML
  - XML Signatures, XBase, XInclude
  - XML Schema Adjuncts
- Java-Based XML Data Processing Frameworks
- Assignment #3 (due next week)
Summary of Previous Session

- Introduction to XML Processing
- SAX Processing
- XML SAX Parsers
- SAX and the JAXP APIs
- XML application Development using the XML Java APIs
- Java-based XML application support frameworks
- Assignment 3a+3b (due in two weeks)

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
Part I

*DOM Processing*

**DOM APIs**

- Document Object Model (DOM) API
  - DOM API defines a minimal set of language/platform independent interfaces for accessing/manipulating XML data in a hierarchical manner
  - Tree structure-based API
    - 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
  - See [http://www.developerlife.com/domintro/default.htm](http://www.developerlife.com/domintro/default.htm)
  - Issued as a W3C recommendation (10/98)
  - See Session 6 Sub-Topic 1 Presentation
- Pure Java APIs: JDOM (Open Source) and JAXP
  - Open Source Pure Java API (JDOM)

See Session 6 Sub-Topic 1 Presentation
DOM Representation of an XML Document


XML Parser and DOM Layer

Figure 1: XML Parser and DOM Layer
DOM Object
Hierarchical Structure

![XML Document and Document Object Tree]

Figure 2: Hierarchical structure of a document object

DOM Inheritance Relationships

![Diagram showing inheritance relationships between Node, Element, and Document interfaces]

The main interface in DOM is Node. Almost every interface in DOM is a subclass of Node. The other interfaces are somewhat redundant and are provided for a more object-oriented access to certain types of Node objects, like Element and Document objects.

Figure 3: Inheritance relationships between the important DOM interfaces
DOM Document Object Tree

Accessing Children of a Node

Children Nodes are contained in a NodeList object. You can ask a Node of type ELEMENT_NODE to give you a list of its children Nodes (if it has any). Those Nodes are contained in a NodeList object. The NodeList object allows you to iterate every Node object contained in it.
Characteristics of a DOM Node

- Node Type, Tag Name, and Value
  - e.g., `<email> xml@beanfactory.com </email>`
- Node’s Children Nodes
  <person>
  <name> Nazmul Idris </name>
  <email> xml@beanfactory.com </email>
  </person>
- If a Node object has a value (i.e. it contains some text data), it may or may not have children elements.
- If a Node object has children, it may or may not have a value
- A Node may also have no children or no value

Using DOM

- DOM interfaces exist in org.w3c.dom
  - Use `import org.w3c.dom.*;`
- Implementation of DOM interfaces are provided by XML parsers
- Corresponding Implementation jar files must be listed in CLASSPATH
DOM Standards
http://www.w3.org/DOM/

- W3C DOM Activity Statement
  - http://www.w3.org/DOM/Activity
- Working Drafts
  - DOM 3 Load and Save (2/26/03)
  - DOM 3 Core (http://www.w3.org/TR/2003/WD-DOM-Level-3-Core-20030609/)
  - DOM 3 XPath, Events, and Validation
- Public Release of Specifications
  - http://www.w3.org/DOM/DOMTR
  - http://www.w3.org/TR/DOM-Level-2-Core/
- JAXP 1.2
  - Includes APIs for processing XML documents using SAX, DOM, and XSLT
  - XML Schema and XSLT compiler (XSLTC) support are new features in 1.2

DOM Evolutions
http://www.ibiblio.org/xml/books/xmljava/chapters/ch09.html#d0e13336

- DOM Level 0
  - Applied to HTML documents in the context of JavaScript
- DOM Level 1
  - Addresses incompatibilities between browser models
- DOM Level 2
  - Cleaned up version of DOM Level 1 interfaces
  - Namespace support in Element and Attr interfaces
  - Additional interfaces
    - Events, traversal, ranges, views, style sheets
- DOM Level 3
  - Full support of all Infoset properties
  - Parser independent creation of a new Document object
  - Added support for DTDs and Schemas
DOM 3 Improvements


- Ability to “bootstrap” a DOM implementation without vendor-specific code
  - Allows applications to run with various parsers
  - In DOM 2, you must either:
    - Instantiate Document implementation and get DOMImplementation from it
    - Instantiate a vendor’s DOMImplementation and get a org.w3c.dom.Document object from it
      DOMImplementation domImpl = new org.apache.xerces.dom.DOMImplementationImpl();
  - DOM 3:
    import org.w3c.dom.Document; import org.w3c.dom.DOMImplementation;
    import org.w3c.dom.DOMImplementationFactory;
    DOMImplementation domImpl = DOMImplementationFactory.getDOMImplementation();
    Document doc = domImpl.createDocument();

- Expanded node comparisons capabilities
  - DOM 2 only allows structural equality tests
  - isSameNode(Node input) vs. equalsNode (Node input, boolean deep)

- Full support for XML document items
  - e.g., XML declaration
  - getVersion(), getEncoding(), etc.

DOM Coding Problems


- Too many lines of code required to do simple things
  - e.g., checking the value of an attribute (16 vs. 3 lines!)

- Traversing the DOM
  Element root = doc.getDocumentElement();
  Node configNode = root.getFirstChild();
  NodeList childNodes = configNode.getChildNodes();
  for (int childNum = 0; childNum < childNodes.getLength(); childNum++) {
    if (childNodes.item(childNum).getNodeType() == Node.ELEMENT_NODE ) {
      Element child = (Element) childNodes.item( childNum ) ;
      if ( child.getTagName().equals( "header" ) ) {
        // Do something with the header
        System.out.print(“Got a header!
”); }}
  }

vs. using DOM 2: “getElementsByTagName ()” as in

NodeList names = someElement.getElementsByTagName("name");
Effective Use of DOM

- Do not use DOM to traverse the document
- Use XPath to find nodes or traverse the document
  - XPath can be used to address, search, and match pieces of an XML document
  - Xalan package supports XPath
  - Sample XPath queries:
    - `/x/y/z, /x/y/*, /x/y[@name='a']`
  - XPath returns only the element nodes (not the intermediate text nodes)
  - See: http://www.zvon.org/xxl/XPathTutorial/General/examples.html
  - XPath spec: http://www.w3.org/TR/xpath
- Use a library or higher-level functions to simplify the use of DOM
  - e.g., findValue, setValue, appendNode

Java-enabled XML Technologies
(review)

- 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
Sample Applications

- XML and Java textbook samples:
  - http://pws.prser.net/Hiroshi.Maruyama/xmlbook/samples4v2/
- Sun Developers DOM Samples
- IBM Developerworks samples
- DOM implementations list:
  - http://www.xmlsoftware.com
- Xerces2-J Samples:
  - http://xml.apache.org/xerces2-j/samples-dom.html
- JAXP Samples:
- Mozilla.org DOM Samples
  - http://www.mozilla.org/docs/dom/samples/

Part II

XML DOM Parsers
DOM vs. SAX
(review)

- Object-based interface vs. Event-based interface
- Object model created automatically vs. created by application
- Order/sequencing of the elements preserved vs. ignored in favor of single events
- Higher use of memory vs. lower use of memory
- Slower speed of initial data retrieval vs. faster speed of initial data retrieval
- Better for complex structures vs. better for simple structures
- Both support optional validation via an API in the DOMParser/SAXParser classes
- DOM has the ability to update XML documents

DOM Parsers
http://www.cafeconleche.org/books/xmljava/chapters/ch09s05.html

- Main contenders:
  - Crimson, Xerces, XML for Java, the Oracle XML Parser for Java, and GNU JAXP
  - DOM 2 does not specify how to parse a document, create a new document, or serialize a document into a file/stream
  - Most parsers that support DOM Level 2 also support JAXP 1.2 (part of Java 1.4)
  - DOM Level 3 is not supported by all parsers
  - Speed and memory consumption are parser-specific
    - In-memory trees are 3 to 10 times larger than XML doc
    - Lazy DOM implementation only reads requested parts
DOM Parsers
Memory Utilization

- Size of Document Objects created for a Sample XML Document (197K) + DTD (56K)
  - Xerces-J 2.0.1: 1489K
  - Crimson 1.1.3 (JDK 1.4 default): 1230K
  - Oracle XML Parser for Java 9.2.0.2.0: 2500K

Sample XML parsers and engines

- Mainstream Java XML parsers
  - IBM’s XML for Java
  - Apache’s Xerces-J
  - Java Project X Parser (Crimson)
  - Oracle XML Parser
  - GNU JAXP

- Other Parsers:
  - BXI
  - Dan Connolly’s XML parser
  - XML-Toolkit
  - LTXML
  - Expat
  - TCLXML
  - Sparse
  - XP
  - DataChannel XPLparser (DXP)
  - XML/Parse
  - PyXMLTok
  - Lruk
  - Microsoft’s XML parser
  - Aedred
  - Xmlproc
  - xmlsh
  - OpenXML Parser,
Comprehensive List of XML DOM Processors

- A comprehensive list of parsers is available at http://www.xmlsoftware.com/parsers
  - Includes links to latest product pages
  - Includes Version numbers, Licensing information, and Platform details

Xerces-J Architecture
http://www.apache.org/~andyc/xml/present/Xerces2.ppt

- Goal is a Generic Pipeline configuration:

   XML → Scanner → Validator → Parser → API
Xerces1-J Limitations

- Hard-coded dependencies on implementation
- Inconsistent Interfaces

![Diagram showing XML, Scanner, Validator, Parser, and API with dependencies and different interfaces.]

Xerces2-J is based on XNI

- XNI
  - “Streaming” Information Set
  - Similar to SAX
  - No loss of document information*
- Parser configuration and layering
- Future extensions
  - Native pull-parser, tree model, etc.

* Does not preserve all document information but communicates more information to the application than DOM or SAX.
Xerces2-J XNI Interfaces/Classes

- XMLDocumentHandler
- XMLDTDHandler
- XMLDTDContentModelHandler
- XMLDocumentFragmentHandler
- XMLLocator
- XMLAttributes
- QName
- XMLString
- XMLResourceIdentifier
- XNIException
- java.lang.RuntimeException
- XMLComponentManager
- XMLComponent
- XMLParserConfiguration
- XMLPullParserConfiguration
- XMLDocumentFilter
- XMLDocumentSource
- XMLDocumentScanner
- XMLDTDFilter
- XMLDTDSource
- XMLDTDFilter
- XMLDTDContentModelSource
- XMLDTDFilter
- XMLDTDContentModelSource
- XMLErrorHander
- XMLEntityResolver
- XMLParserConfigurationException
- XMLParseException
- XMLInputSource

Xerces2-J Parser Pipeline

- Handlers communicate information between parser components

XML → Scanner → Validator → Parser → API
Xerces2-J Parser Handler

XML

- Document Scanner
- DTD Scanner
- Validator
- Parser

XMLDocumentHandler
XMLDTDHandler
XMLDTDContentModelHandler

Xerces2-J Parser Layout

- Components and Manager

Component Manager

- Symbol Table
- Grammar Pool
- Datatype Factory
- Scanner
- Validator
- Entity Manager
- Error Reporter

Regular Components
Configurable Components
Xerces2-J Parser Configuration

- Before

  - DOM Parser
  - SAX Parser

  Document Parser

  - Scanner
  - Validator

  XML

* Parser pipeline is part of the document parser base class.

* Required duplication to re-configure parser and still take advantage of API generator code.

Xerces2-J Parser Configuration

- After

  - DOM Parser
  - SAX Parser

  Document Parser

  - Parser Configuration

  - Scanner
  - Validator

  XML

* Parser pipeline and settings are specified in a separate parser configuration object.

* Allows re-use of framework without rewriting existing code.
Xerces2-J Parser API Generators

- Different APIs can be generated from same document parser

Xerces2-J Sample Parser Configuration

- Non-validating parser (for performance)
  - Available with Xerces download
Part III

DOM and the JAXP APIs
(review)

DOM Parsing APIs
DOM API Packages

- **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
XSLT API Packages

- See Session 3 handout on “Processing XML Documents in Java Using XPath and XSLT”
- **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)
  - javax.xml.parsers
  - javax.xml.transform
    - javax.xml.transform.dom
    - javax.xml.transform.sax
    - javax.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)

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)
Part IV

Advanced XML Parser Technology

XML Data Processing Patterns

- Processing and Transformation
  - Reading XML Documents
  - Working with Encodings in XML Documents
- XML Document to/from DOM (generation)
  - Printing XML Documents from DOM
- Building and Working with DOM Structures
  - Creating a DOM Structure from Scratch
  - Building a Valid DOM Tree
  - Manipulations Using DOM API
**JDOM**

- JSR-102
- Beta 6 Available
- Lightweight API with best of SAX and DOM
  - Small memory footprint
  - Does not require whole document in memory
  - Easy to use
    - JDOM: `String text = element.getText();`
    - DOM: `String content = element.getFirstChild().getValue();`
- Converting from DOM to JDOM
  - Use `org.jdom.input.DOMBuilder` class

**JDOM vs. DOM**

- DOM (PL independent, strict hierarchies, interface driven)
  - Language neutrality prevents the use of Java methodologies and patterns
    - e.g., create new Elements via factory constructor methods
    - e.g., collections of Nodes are NodeLists instead of List or Iterator objects
  - Requires more code and is not intuitive to Java programmers
  - Everything-is-a-node but access methods are not uniform for all nodes
    - Values of “CharacterData” nodes are set via “insertData”, while value of “Attr” nodes are set via “setValue”
**JDOM vs. DOM**

(continued)

- **JDOM** (java prone, no hierarchies, class driven)
  - Provides a more natural and easy to use interface
    - Uses native Java representations and objects
    - Provide convenience functions for common tasks
  - Defines node types as separate Java classes
    - E.g., Document, Element, Attribute
    - Developers can use “new” and can limit the need for frequent type casts
  - JDOM strings are represented as Java Strings
  - Collections of nodes are represented via List and Iterator classes

---

**JAXP**

- **JAXP 1.1/1.2**
  - Vendor-neutral code for parsing/transforming documents
  - Updated support for SAX 2.0 & DOM Level 2 standards
  - Addition of TraXP
  - DOM 2 usage patterns:
    - Manipulate Documents, Elements, and Nodes (DOM 1)
    - Views, Stylesheets, Events, traversal, DOM
Parsers Comparison

- See: http://www.webreference.com/xml/column22/2.html
- Apache Xerces supports XML-Schema, XSL-T, SAX 2.0, DOM Level 2 1.0, and is open source
- The Organization for Advancement of Structured Information Systems (OASYS) has defined an XML conformance test suite
  - Sun’s Parser passes all the tests
  - Oracle’s v2 XML parser is most efficient

Part V

Latest W3C APIs and Standards for Processing XML
XML Infoset

- Candidate recommendation (May 14, 2001)
- XML Information Set is a work in progress
- Provides a means to describe the abstract logical representation of an XML document
- Infoset’s prime citizens are information items
  - Information items can be used to express the data model of XML documents, and DTDs
- Not an API

DOM Level 3

- W3C Working Draft (June 5, 2001)
  - Focus on Platform and language neutral interface to DOM
Canonical XML

- W3C Recommendation (March 15, 2001)
- Canonical XML will be used to represent the result of parsing an XML document
- Canonical XML is necessary to establish the logical “equivalence” of XML documents
- Every well-formed XML document has a unique structurally equivalent canonical XML document
- When “Canonicalizing” an XML document, parsers preserves the minimum information needed by an XML application

XML Signatures

- Joint W3C and IETF Working Group
- W3C Candidate Recommendation (April 19, 2001)
- XML compliant syntax used for representing the signature of Web resources and portions of protocol messages
- Procedures for computing and verifying such signatures (I.e., integrity and authentication)
- Does not address encryption, and authorization
XBase

- W3C Working Draft (February 21, 2001)
- `xml:base` may be inserted in XML documents to specify a base URI other than the base URI of the document or external entity, which is normally used to resolve relative URIs
- Equivalent of HTML BASE functionality

XInclude

- W3C Working Draft (May 16, 2001)
- Processing model and syntax for merging multiple Infosets in a single “composite” Infoset
- Enables modularity and merging capabilities
XML Schema Adjuncts

- Mechanism for extending XML schema languages, and for providing information from such extensions to programs that process XML instances
- Schema Adjunct Framework
  - XML-based language used to associate domain-specific data (i.e., adjunct-data) with schemas and their instances, effectively extending the power of existing XML schema languages such as DTDs or XML Schema
  - Domain specific data include O/R mappings, etc.

Part VI

Java-Based XML Data Processing
Frameworks and APIs
Applets, Servlets, and JSPs

- These component models provide infrastructure support for XML processing
- See Session 6 handout on “Applets, Servlets, and JSPs”

Xerces-J

- Xerces 2 is a redesigned implementation of Xerces that emphasizes modularity
- Xerces 2’s architecture is based on the Xerces Native Interface (XNI) which streams the processing of XML documents via “Scanner”, “Validator” and “Parser” modules
- A parser configuration object controls the use of various internal components (e.g., symbol table, validator, etc.)
**Xalan**

- Xalan-J is the Java implementation
- Provides XSL-T processing for transforming XML documents into HTML, text, or other XML document types
- Built on top of SAX 2.0, DOM Level 2 1.0, JAXP 1.1
- Implements the TraX subset of JAXP 1.1

---

**Xang**

- Framework for building data-driven, cross-platform Web applications that integrate disparate data sources
- Separates data, logic and presentation
- Example .xap command handler
  - `<xap onGet='doGet();' onGetPurchase='run_purchase();'>`
- **Xang:**
  - Identifies element addressed by URLs
  - Map the HTTP request to an application method on the targeted element
  - Gather a body of script available to the command handler for the targeted element
  - Dispatch the HTTP request to the command handler
FOP

- Latest version is 0.19
- Print formatter driven by XSL-FO objects
- Formatted output is in PDF format for now
- Can be embedded in a Java application by instantiating `org.apache.fop.apps.Driver`

Cocoon

- Cocoon 2 is a completely redesigned version
- It supports an event-based model, and manages memory very efficiently
SOAP

- Lightweight protocol for information exchange
  - Envelope
    - Framework that describes a message and its processing
  - Set of encoding rules
    - Used to express instances of application datatypes
  - Convention
    - RPC calls and responses
- Apache distributes an implementation of SOAP referred to as “AXIS”

Batik

- Provides a set of core modules to support SVG solutions
  - SVG parsers
  - SVG generators
  - SVG implementations
Crimson

- Implements JAXP 1.1 without the java.xml.transform package
- Supports SAX 2.0, and DOM Level 2 1.0
- Xalan-J 2 implements the missing package
- Based on Sun Project X parser
- Will move under Xerces-J 2

Part VII

Conclusions
Summary

- DOM and SAX Parsers support a common set of XML data processing patterns
- JDOM is a lightweight API that supports the best of DOM and SAX
- JAXP 1.1 provides parser-independent interfaces
- Important comparison criteria for parsers are the support of the latest W3C specification, conformance, and efficiency
- The latest W3C APIs and standards for processing XML have not reached the level of full recommendations and are not embedded in the functionality supported by parsers
- Various specialized XML processing frameworks are being developed by the Apache software foundation. These frameworks are processing engines that leverage off of parsing and rendition mechanisms, and operate on top of common Java components models.

Readings

- Readings
  - XML and Java: Chapter 4, 6, Appendices A-C
  - Processing XML with Java: Chapters 5, 9-15, Appendix A
  - Handouts posted on the course web site
  - Review XSL (XSLT, and XSL-FO) W3C Recommendations
- 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 (or BES 5.2), WebLogic 6.1-8.1, WAS 5.0
  - POSE & KVM (See Session 3 handout)
Assignment

- Assignment #3:
  - Continued from Session 5

Next Session:
XML Information Rendering (Part I)

- Extensible Stylesheet Language Transformation (XSLT)
- Extensible Stylesheet Language Formatting Object (XSL-FO)
- XML and Document /Content Management
- Introduction to XML Application Servers
- Working with XSLT and XSL-FO processors