Overview

- Introduction
  - Uses of XML
  - Content Models
  - Document Object Models
- Parser APIs
  - DOM
  - SAX
- Different Parsers
  - "XML" Parsers
  - "Java" Parsers

Context

- At the heart of the XML family of standards is the notion of a rigorous model of a document
- XML is now being used for a variety of functions
  - XML is being used increasingly as an interchange format.
  - XML is being used as a canonical document format
- All of these uses make it desirable to programatically manipulate XML documents and Parsers are the key to doing this.
The Uses of XML

- Two major roles have emerged for XML:
  - As a language that can more accurately define various specialized kinds of documents
  - As a language that can encapsulate data interchanged between applications
- The XML documents produced for both of these purposes may be envisioned as either:
  - a simple serial byte stream
  - a "directed acyclic graph" – a tree.

Parsers and API's

- A parser is a program or function that receives input in the form of sequential set of instructions or tokens and analyzes them for "correctness":
  - A lexical analyzer builds a sequence of tokens
  - A syntactic analyzer produces abstract syntax trees
- A parser may be:
  - A standalone application
  - A library function with some form of application program interface (API) to control and examine its operation.
  - A parser may support one or more APIs
  - The API(s) to a parser may be standard or idiosyncratic.

Patterns and APIs

- In building an API, more general patterns may be used as a part of the API.
- Some of those used in Java, dating back to CORBA, include factories and builders.
  - The factory pattern is used to provide access to concrete classes that may be instantiated via a more abstract class.
  - The builder pattern is used to create complex objects, such as documents.
  - The observer pattern is used for the various interfaces and adapters.
**XML Parsers**

- The XML standard defines the role of XML parsers.
- Thus we speak of XML parsers as being conformant or non-conformant.
- The XML standard defines two major roles for the parser:
  - Determining well-formedness
  - Determining validity
- In order for an XML parser to determine validity, it must be able to assess the DTD or schema associated with the document.

**Complications of Parsing**

- Checking a document for well-formedness can be complicated.
  - The rules for XML syntax were simplified to make parsing easier.
- Checking a document for validity can be VERY complicated.
  - The parser must also be able to parse a DTD.
  - DTD’s were replaced by schema to “ease” this burden
  - Schema quickly became more complex making this more difficult yet

**Complications of Parsing (Cont.)**

- Some of the pragmatic complications include:
  - Is white space preserved?
  - How should CDATA sections be dealt with
  - How should notations be dealt with
  - How do DTDs and Namespaces get loaded?
  - Do entity definitions get checked?
  - Do entities get replaced?
  - How do entities in DTDs and namespaces get resolved?
  - Do data types get checked?
  - How are errors reported?
Other Variations

- There are a number of Parsers available with different capabilities. These include Xerces-J, Crimson, Aelfred, MSXML3. They differ in functionality – namespaces, interfaces, etc.
- JAXP is the Java API bundled with Java 1.4-1.6. In addition to SAX and DOM it includes TrAX and some factory and builder functions.
  - JDOM, dom4j, and ElectricXML are APIs to the document tree that is built for general consistency with Java
  - SAX is a stream API that makes calls on the parser rather than being called by the parser
  - TrAX is an interface that uses transformers based on XSLT and XPath.

Streams and Trees

- All XML documents begin as a byte stream and this is where parsers must begin.
- We think of XML documents as trees, branches, and sets of nodes
- The presentation of the parsing to an application – the Application Program Interface (API) to the parser, may present either access to the byte stream view, access to the tree view, or some intermediate view.
- Two more or less standard API views have emerged:
  - The Simple API for XML (SAX) which operates on a byte stream
  - The Document Object Model (DOM) API which operates on the tree
- While the APIs are becoming more stable, they are still evolving

SAX and DOM

- SAX is the most standard API for XML parsing and it is unique in that it is modeled after the parsing process rather than the XML tree
- DOM is the first of the tree model parsers and returns a complete XML document as a tree
Simple API for XML (SAX)

- The SAX is a very lightweight approach to scanning XML documents.
- SAX checks for document well-formedness and can also check for validity.
- SAX is very efficient and fast – allowing files of any size to be processed.
- SAX provides access to one element at a time allowing the user to build their own data structure.
- It is generally not used for changing documents or creating them – simply for reading them.

Document Object Model (DOM) API

- The DOM API is a heavyweight approach to XML documents.
- DOM checks for document well-formedness and can also check for validity.
- DOM converts the XML document to a tree which is held in memory (it is memory intensive).
- The DOM API:
  - Allows manipulation of the tree.
  - Will convert the tree to a serial stream (file, socket, or byte stream).