Overview

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

Context

- SGML was designed as a way to represent classes of structured documents.
- HTML was defined as an SGML document type that would allow for document interchange over the web.
- With the growth of the web, and e-business, HTML was found to be too limited – more document types were needed.
- For a variety of reasons, SGML DTD parsing was considered too complicated and XML was developed as a simplified form of SGML.
- XML is being used increasingly as an interchange format.
The Uses of XML

• XML, whether it is used to encapsulate simple data records or complex documents, may be envisioned as either a byte stream or as a “directed acyclic graph” – a tree.
• Two 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

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
• The parser has 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?
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.

<table>
<thead>
<tr>
<th>Parser</th>
<th>API Support</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Xerces-J</td>
<td>SAX2, DOM2, JAXP</td>
<td>NS Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Schema Partial</td>
</tr>
<tr>
<td>Crimson</td>
<td>SAX2, DOM2, JAXP</td>
<td>NS Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Schema no</td>
</tr>
<tr>
<td>AElfred</td>
<td>SAX2, DOM2 – different parsers</td>
<td>NS yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Schema no</td>
</tr>
<tr>
<td>Piccolo</td>
<td>SAX2</td>
<td>Non - validating</td>
</tr>
</tbody>
</table>

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 converts the tree to a serial stream (file, socket, or byte stream).

Other Variations

- JAXP is the Java API bundled with Java 1.4. In addition to SAX and DOM it includes TrAX and some factory and builder functions.
  - TrAX is an interface that uses transformers based on XSLT and XPath.
- JDOM, dom4j, and ElectricXML are APIs to the document tree that is built for general consistency with Java.
- XMLPULL is a stream API that makes calls on the parser rather than being called by the parser.