3. XML Processor APIs

- How can applications manipulate structured documents?
  - An overview of document parser interfaces

3.1 SAX: an event-based interface
3.2 DOM: an object-based interface
3.3 JAXP: Java API for XML Processing

Tasks of a Parser

- Document instance decomposition
  - elements, attributes, text, processing instructions, entities, ...
- Verification
  - well-formedness checking
  - syntactical correctness of XML markup
  - validation (against a DTD or Schema)
- Access to contents of the DTD (if supported)
  - SAX 2.0 Extensions provide info of declarations: element type names and their content model expressions

Document Parser Interfaces

I: Event-based interfaces
  - Command line and ESIS interfaces
    - Element Structure Information Set, traditional interface to stand-alone SGML parsers
    - Event call-back interfaces: SAX
II: Tree-based (object model) interfaces
  - W3C DOM Recommendation

Command-line ESIS interface

- Application
  - Command line call
  - SGML/XML Parser
  - ESIS Stream
  - `<A i="1">Hi!</A>`

Event Call-Back Interfaces

- Application implements a set of callback methods for handling parse events
  - parser notifies the application by method calls
  - method parameters qualify events further
    - element type name
    - names and values of attributes
    - values of content strings, ...
- Idea behind "SAX" (Simple API for XML)
  - an industry standard API for XML parsers
  - could think as "Serial Access XML"
An event callback application

- Application Main Routine
  - `startDocument()`
  - `startElement()`
  - `characters()`
  - `endElement()`

Example:
```
<xml version="1.0"/>
<A i="1">Hi!</A>
```

Object Model Interfaces

- Application interacts with
  - a parser object
  - a document parse tree object consisting of objects for `document`, `elements`, `attributes`, `text`, ...

- Abstraction level higher than in event based interfaces; more powerful access
  - to descendants, following siblings, ...

- Drawback: Higher memory consumption
  - > applied mainly in client applications
    (to implement document manipulation by user)

An object-model based application

- Application
- Access/Modify
- In-Memory
- Document
- Representation

Example:
```
<A i="1">Hi!</A>
```

3.1 The SAX event callback API

- A de-facto industry standard
  - Developed by members of the xml-dev mailing list
  - Version 1.0 in May 1998, Ver. 2.0 in May 2000
  - Not a parser, but a common interface for many different parsers (like, say, JDBC is a common interface to various RDMS)

- Supported directly by major XML parsers
  - most Java based and free:
    - Apache Xerces, Oracle’s XML Parser for Java;
    - MSXML (in IE 5), James Clark’s XP

SAX 2.0 Interfaces

- Interplay between an application and a SAX-conformant parser specified in terms of `interfaces` i.e., collections of methods

- One way to classify of SAX interfaces:
  - Parser-to-application (or callback) interfaces
    - to attach special behaviour to parsing events
  - Application-to-parser interfaces
    - to use the parser
  - Auxiliary interfaces
    - to manipulate parser-provided information

Call-Back Interfaces

- Implemented by application to override default behaviour (of ignoring events quietly)
  - `ContentHandler`
    - methods to process document parsing events
  - `DTDHandler`
    - methods to receive notification of unparsed external entities and their notations declared in the DTD
  - `ErrorHandler`
    - methods for handling parsing errors and warnings
  - `EntityResolver`
    - methods for customised processing of external entity references
Application-to-Parser Interfaces

- Implemented by parser (or a SAX driver):
  - XMLReader
    - methods to invoke the parser and to register objects that implement call-back interfaces
  - XMLFilter (extends XMLReader)
    - interface to connect XMLReaders in a row as a sequence of filters
    - obtains events from an XMLReader and passes them further (possibly modified)

The ContentHandler Interface

- Information of general document events. (See API documentation for a complete list)
- setDocumentLocator(Locator locator)
  - Receive a Locator for the origin of SAX document events
- startDocument(); endDocument()
  - notify the beginning/end of a document
- startElement(String nsURI, String localName, String rawName,Attributes atts);
  - similar (without attributes)

Namespaces in SAX: Example

```xml
<stylesheet version="1.0"
xmlns:xsl=http://www.w3.org/1999/XSL/Transform
xmlns=http://www.w3.org/TR/xhtml1/strict">
  <xsl:template match="/">
    <html>...
  </xsl:template>
</stylesheet>
```

Namespaces: Example (2)

```xml
<stylesheet version="1.0"
xmlns:xsl=http://www.w3.org/1999/XSL/Transform
xmlns=http://www.w3.org/TR/xhtml1/strict
xmlns:html=http://www.w3.org/TR/xhtml1/strict

<xsl:stylesheet>
  <xsl:template match="/">
    <html>...
  </xsl:template>
</xsl:stylesheet>
```

ContentHandler interface (cont.)

- characters(char ch[], int start, int length)
  - notification of character data
- ignorableWhitespace(char ch[], int start, int length)
  - notification of ignorable whitespace in element content

```xml
<doc-type A '{<element A (B)>}'>
  <A>
    <B>
      Text content
    </B>
  </A>
</doc-type>
```
SAX Processing Example (1)

- **Input:** XML representation of a personnel database:

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<db>
  <person idname="1234">
    <lastName>Kilpeläinen</lastName>
    <firstName>Pekka</firstName>
  </person>
  <person idname="5678">
    <lastName>Määttänen</lastName>
    <firstName>Matti</firstName>
  </person>
  <person idname="9012">
    <lastName>Möttönen</lastName>
    <firstName>Maija</firstName>
  </person>
  <person idname="3456">
    <lastName>Röppänen</lastName>
    <firstName>Maija</firstName>
  </person>
</db>
```

SAX Processing Example (2)

- **Task:** Format the document as a list like this:

```
Pekka Kilpeläinen (1234)
Matti Määttänen (5678)
Maija Möttönen (9012)
Maija Röppänen (3456)
```

- **Event-based processing strategy:**
  - At the start of a person, record the idnum (e.g., 1234)
  - Notify starts and ends of last and first to record their contents (e.g., "Kilpeläinen" and "Pekka")
  - At the end of a person, output the collected data

SAX Processing Example (3)

- **Application:** Begin by importing relevant classes:

```java
import org.xml.sax.XMLReader;
import org.xml.sax.Attributes;
import org.xml.sax.ContentHandler;
//Default (no-op) implementation of
//interface ContentHandler:
import org.xml.sax.helpers.DefaultHandler;
//SUN JAXP to instantiate a parser:
import javax.xml.parsers.*;
```

SAX Processing Example (4)

- **Implement relevant call-back methods:**

```java
public class SAXDBApp extends DefaultHandler{
    // Flags to remember element context:
    private boolean InFirst = false,
    InLast = false;
    // Storage for element contents and
    // attribute values:
    private String FirstName, LastName, IdNum;
```

SAX Processing Example (5)

- **Call-back methods:**
  - Record the start of first and last elements,
  - and the idnum attribute of a person.

```java
public void startElement( 
    String namespaceURI, String localName,
    String rawName, Attributes atts) { 
    if (rawName.equals("first")) 
        InFirst = true;
    if (rawName.equals("last"))
        InLast = true;
    if (rawName.equals("person"))
        IdNum = atts.getValue("idnum");
} // startElement
```

SAX Processing Example (6)

- **Call-back methods continue:**
  - Record the text content of elements first and last:

```java
public void characters ( 
    char[] ch, int start, int length) { 
    if (InFirst) FirstName = 
        new String(ch, start, length);
    if (InLast) LastName = 
        new String(ch, start, length);
} // characters
```
SAX Processing Example (7)

- At the end of `person`, output the collected data:

```java
public void endElement(String namespaceURI, String localName, String qName) {
    if (qName.equals("person")) {
        System.out.println(FirstName + " " + LastName + " " + IdNum + ");
        // Update the context flags:
        if (qName.equals("first"))
            InFirst = false;
        // (Similarly for "last" and InLast)
```