3.3 JAXP: Java API for XML Processing

- How can applications use XML processors?
  - A Java-based answer: through JAXP
  - An overview of the JAXP interface
    - What does it specify?
    - What can be done with it?
  - How do the JAXP components fit together?

(Partly based on tutorial "An Overview of the APIs" available at http://java.sun.com/xml/jaxp/dtd/1.1/docs/tutorial/overview/3_api.html, from which also some graphics are borrowed)

JAXP 1.1

- An interface for "plugging-in" and using XML processors in Java applications
  - Includes packages
  - org.xml.sax: SAX 2.0 interface
  - org.w3c.dom: DOM Level 2 interface
  - javax.xml.parsers: initialization and use of parsers
  - javax.xml.transform: initialization and use of transformers (XSLT processors)
  - Included in JDK starting from vers. 1.4

JAXP: XML processor plugin (1)

- Vendor-independent method for selecting processor implementation at run time
  - Principally through system properties
    - java.xml.parsers.SAXParserFactory
    - java.xml.parsers.DocumentBuilderFactory
    - java.xml.transform.TransformerFactory
  - Set on command line (to use Apache Xerces as the DOM implementation):
    - java

JAXP: XML processor plugin (2)

- Set during execution (-> Saxon as the XSLT impl):
- By default, reference implementations used
  - Apache Xerces/Xerces-2 as the XML parser
  - Apache Xalan as the XSLT processor
- Currently supported only by a few compliant XML processors:
  - Parsers: Apache Xerces and Xerces-2, Aelfred, Oracle XML Parser for Java
  - XSLT transforms: Apache Xalan, Saxon

JAXP: Functionality

- Parsing using SAX 2.0 or DOM Level 2
- Transformation using XSLT
  - (We'll study XSLT in detail later)
- Fixes features left unspecified in SAX 2.0 and DOM 2 interfaces
  - control of parser validation and error handling
    - error handling behavior can be controlled in SAX by implementing ErrorListener methods
  - loading and saving of DOM Document objects

JAXP Parsing API

- Included in JAXP package
  - javax.xml.parsers
- Used for invoking and using SAX
  - SAXParserFactory spf = SAXParserFactory.newInstance();
  - and DOM parser implementations:
    - DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
JAXP: Using a SAX parser (1)

We have already seen this:
SAXParserFactory spf = SAXParserFactory.newInstance();
try {
    SAXParser saxParser = spf.newSAXParser();
    XMLReader xmlReader = saxParser.getXMLReader();
    } catch (Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
    xmlReader.setContentHandler(handler);
    xmlReader.parse(filename);

JAXP: Using a SAX parser (2)

JAXP: Using a DOM parser (1)

Code to parse a file into DOM:
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
try {
    DocumentBuilder builder = dbf.newDocumentBuilder();
    } catch (ParserConfigurationException e) {
        e.printStackTrace();
        System.exit(1);
    }
    Document domDoc = builder.parse(filename);

JAXP: Using a DOM parser (2)

DOM building in JAXP

Errors of DOM parsing can be handled
- by creating a SAX ErrorHandler, say errorHandler
  to implement error, fatalError and warning methods
  and passing it to the DocumentBuilder (before parsing):
  builder.setErrorHandler(errorHandler);
- Parser properties can be configured:
  - for both SAXParserFactories and
    DocumentBuilderFactories (before parser creation):
      factory.setValidating(true/false)
      factory.setNamespaceAware(true/false)
JAXP: Controlling parsing (2)

- Further DocumentBuilderFactory configuration methods to control the form of the resulting DOM tree:
  - setIgnoringComments (true/false)
  - setIgnoringElementContentWhitespace (true/false)
  - setCoalescing (true/false)
- combine CDATA sections with surrounding text?
  - setExpandEntityReferences (true/false)

JAXP Transformation API

- Earlier known as TrAX
- Allows application to apply a Transformer to a Source document to get a Result document
- Transformer can be created:
  - from XSLT transformation instructions (to be discussed later)
  - without instructions
    - gives an identity transformation, which simply copies the Source to the Result

JAXP: Using Transformers (1)

JAXP Transformation Packages

- javax.xml.transform:
  - Classes Transformer and TransformerFactory; initialization similar to parsers and parser factories
- Transformation Source object can be:
  - a DOM tree, a SAX XMLReader or an input stream
- Transformation Result object can be:
  - a DOM tree, a SAX ContentHandler or an output stream

Source-Result combinations

JAXP Transformation Packages (2)

- Classes to create Source and Result objects from DOM, SAX and I/O streams defined in packages
  - javax.xml.transform, javax.xml.transform.sax, and javax.xml.transform.stream
- An identity transformation from a DOM Document to I/O stream gives a vendor-neutral way to serialize DOM documents
  - (the only option in JAXP)
Serializing a DOM Document as XML text

- Identity transformation to an output stream:

```java
TransformerFactory tFactory = TransformerFactory.newInstance();
// Create an identity transformer:
Transformer transformer = tFactory.newTransformer();
DOMSource source = new DOMSource(myDOMdoc);
StreamResult result =
    new StreamResult(System.out);
transformer.transform(source, result);
```

Controlling the form of the result?

- As above, but create a Transformer with XSLT instructions as a StreamSource, say `saveSpecSrc`:

```xml
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:output encoding="ISO-8859-1" indent="yes"
        doctype-system="reglist.dtd" />
    <xsl:template match="/"> <!-copy entire doc--> <xsl:copy-of select="." />
    </xsl:template>
</xsl:stylesheet>
```

// Now create a tailored transformer:
Transform transformer =
    tFactory.newTransformer(saveSpecSrc);

Other Java APIs for XML

- JDOM
  - a Java-specific variant of W3C DOM
  - http://www.jdom.org/

- DOM4J (http://www.dom4j.org/)
  - roughly similar to JDOM; richer set of features:
  - powerful navigation with integrated XPath support

- JAXB (Java Architecture for XML Binding)
  - compiles DTDs to DTD-specific classes for reading, manipulating and writing valid documents
  - http://java.sun.com/xml/jaxb/

JAXP: Summary

- An interface for using XML Processors
  - SAX/DOM parsers, XSLT transformers
- Supports pluggability of XML processors
- Defines means to control parsing and handling of parse errors (through SAX ErrorHandlers)
- Defines means to write out DOM Documents
- Included in JDK 1.4