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/dist/1.1/docs/tutorial/overview/3_apis.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
  - For example:

JAXP: XML processor plugin (2)

- By default, reference implementations used
  - Apache Crimson/Xerces as the XML parser
  - Apache Xalan as the XSLT processor
- Currently supported only by a few compliant XML processors:
  - Parsers: Apache Crimson and Xerces, Aelfred
  - XSLT transformers: Apache Xalan, Saxon

JAXP: Functionality

- Parsing using SAX 2.0 or DOM Level 2
- Transformation using XSLT
  - (We'll perform stand-alone transformations later)
- Fixes features left unspecified in SAX 2.0 and DOM Level 2
  - control of parser validation and error handling
  - creation and saving of DOM Document objects

JAXP Parsing API

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

```java
SAXParserFactory spf = SAXParserFactory.newInstance();
try {
    SAXParser saxParser = spf.newSAXParser();
    XMLReader xmlReader = saxParser.getXMLReader();
} catch (Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

JAXP: Using an SAX parser (2)

- We've already used this:

```java
SAXParserFactory spf = SAXParserFactory.newInstance();
try {
    SAXParser saxParser = spf.newSAXParser();
    XMLReader xmlReader = saxParser.getXMLReader();
} catch (Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}
```

JAXP: Using a DOM parser (1)

```java
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
try {
    // to get a new DocumentBuilder:
    DocumentBuilder builder = dbf.newDocumentBuilder();
} catch (ParserConfigurationException e) {
    e.printStackTrace();
    System.exit(1);
}
```

JAXP: Using a DOM parser (2)

- We've used this, too:

```java
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
try {
    // to get a new DocumentBuilder:
    DocumentBuilder builder = dbf.newDocumentBuilder();
} catch (ParserConfigurationException e) {
    e.printStackTrace();
    System.exit(1);
}
```

DOM building in JAXP

- Errors of DOM parsing can be handled
  - by creating a SAX ErrorHandler, which implements `error`, `fatalError` and `warning` methods, and passing it with `setErrorHandler` to the `DocumentBuilder`
- Validation and namespace processing can be controlled, both for `SAXParserFactories` and `DocumentBuilderFactory` with `setValidating(boolean)` and `setNamespaceAware(boolean)`
### JAXP Transformation API

- also 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, which gives an identity transformation (simply copies Source to Result)

### 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, an SAX XMLReader or an I/O stream
- Transformation Result object can be
  - a DOM tree, an SAX ContentHandler or an I/O stream

### Serializing a DOM Document as XML text

- Identity transformation to an I/O stream Result:

  ```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);
  ```

### Other Java APIs for XML

- **JDOM**
  - variant of W3C DOM; closer to Java object-orientation (http://www.jdom.org/)
- **DOM4J** (http://www.dom4j.org/)
  - roughly similar to JDOM; richer set of features
- **JAXB (Java Architecture for XML Binding)**
  - compiles DTDs to DTD-specific classes that allow to read, manipulate and to write valid documents
  - http://java.sun.com/xml/jaxb/
JAXP: Summary

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