

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 Sun tutorial "An Overview of the APIs", from which some graphics are borrowed; currently Chap 4 in online J2EE 1.4 Tutorial]

SDPL 2008

3.3: (XML APIs) JAXP

1

Some History: JAXP Versions

- JAXP 1.1 included in Java JDK 1.4
- An interface for "plugging-in" and using XML processors in Java applications
 - includes packages
 - » `org.xml.sax`: SAX 2.0
 - » `org.w3c.dom`: DOM Level 2
 - » `javax.xml.parsers`: initialization and use of parsers
 - » `javax.xml.transform`: initialization and use of Transformers (XSLT processors)

SDPL 2008

3.3: (XML APIs) JAXP

2

Later Versions: 1.2

- JAXP 1.2 added property-strings for setting the language and source of a *schema* used for validation
 - `http://java.sun.com/xml/jaxp/properties/schemaLanguage`
 - `http://java.sun.com/xml/jaxp/properties/schemaSource`
- in JAXP 1.3 through the `setSchema (Schema)` method of the **Factory** classes (used to initialize SAXParsers or DOM DocumentBuilders)

SDPL 2008

3.3: (XML APIs) JAXP

3

Later Versions: 1.3 & 1.4

- JAXP 1.3 major update, included in JDK 1.5 (2005)
 - more flexible validation (decoupled from parsing)
 - DOM Level 3 Core, and Load and Save
 - API for applying XPath to do documents
 - mapping btw XML Schema and Java data types
- JAXP 1.4 maintenance release, included in JDK 1.6
 - supports the Streaming API for XML (StAX)
- We'll concentrate to basic ideas (of JAXP 1.1)

SDPL 2008

3.3: (XML APIs) JAXP

4

JAXP: XML processor plugin (1)

- Vendor-independent method for selecting processor implementations at run time
 - principally through system properties

```
javax.xml.parsers.SAXParserFactory
javax.xml.parsers.DocumentBuilderFactory
javax.xml.transform.TransformerFactory
```
 - Set on command line (say, to select Xerces (current default) as the DOM implementation):

```
java
-Djavax.xml.parsers.DocumentBuilderFactory=
org.apache.xerces.jaxp.DocumentBuilderFactoryImpl
```

SDPL 2008

3.3: (XML APIs) JAXP

5

JAXP: XML processor plugin (2)

- Set during execution (-> Saxon as the XSLT impl):

```
System.setProperty(
"javax.xml.transform.TransformerFactory",
"com.icl.saxon.TransformerFactoryImpl");
```
- By default, reference implementations used
 - Apache Xerces as the XML parser
 - Xalan (JAXP 1.2) / XSLTC (JAXP 1.3) as the XSLT processor
- Supported by a few compliant processors:
 - Parsers: Apache Crimson and Xerces, Aelfred, Oracle XML Parser for Java, libxml2 (via GNU JAXP libxmlj) "highly experimental"
 - Transformers: Apache Xalan, Saxon, GNU XSL transformer

SDPL 2008

3.3: (XML APIs) JAXP

6

JAXP: Basic Functionality

- Parsing using SAX 2.0 or DOM Level 2
- Transformation using XSLT
 - (more about XSLT later)
- Adds functionality missing from SAX 2.0 and DOM Level 2:
 - controlling validation and handling of parse errors
 - » error handling **can** be controlled in SAX, by implementing ErrorHandler methods
 - loading and saving of DOM Document objects

SDPL 2008

3.3: (XML APIs) JAXP

7

JAXP Parsing API

- Included in JAXP package `javax.xml.parsers`
- Used for invoking and using SAX ...

```
SAXParserFactory spf =
SAXParserFactory.newInstance();
```

and DOM parser implementations:

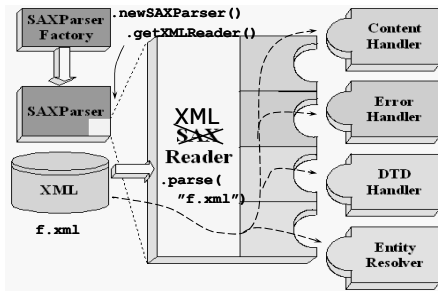
```
DocumentBuilderFactory dbf =
DocumentBuilderFactory.newInstance();
```

SDPL 2008

3.3: (XML APIs) JAXP

8

JAXP: Using a SAX parser (1)



SDPL 2008

3.3: (XML APIs) JAXP

9

JAXP: Using a SAX parser (2)

■ We have already seen this:

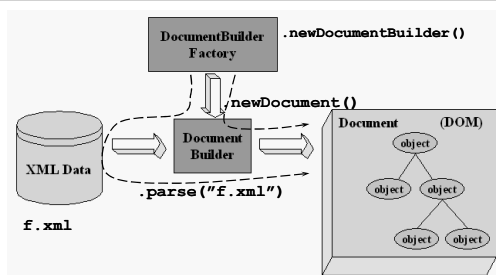
```
SAXParserFactory spf =
    SAXParserFactory.newInstance();
try { SAXParser saxParser = spf.newSAXParser();
    XMLReader xmlReader =
        saxParser.getXMLReader();
    ContentHandler handler = new myHdlr();
    xmlReader.setContentHandler(handler);
    xmlReader.parse(systemIdOrInputSrc);
} catch (Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
};
```

SDPL 2008

3.3: (XML APIs) JAXP

10

JAXP: Using a DOM parser (1)



SDPL 2008

3.3: (XML APIs) JAXP

11

JAXP: Using a DOM parser (2)

■ Parsing a file into a DOM Document:

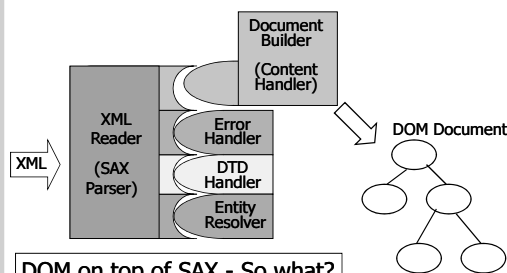
```
DocumentBuilderFactory dbf =
    DocumentBuilderFactory.newInstance();
try { // to get a new DocumentBuilder:
    DocumentBuilder builder =
        dbf.newDocumentBuilder();
    Document domDoc =
        builder.parse(fileNameOrURIetc);
} catch (ParserConfigurationException e) {
    e.printStackTrace();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

SDPL 2008

3.3: (XML APIs) JAXP

12

DOM building in JAXP



SDPL 2008

3.3: (XML APIs) JAXP

13

JAXP: Controlling parsing (1)

- Errors of DOM parsing can be handled
 - by creating a SAX ErrorHandler
 - » to implement error, fatalError and warning methods
 and passing it to the DocumentBuilder:


```
builder.setErrorHandler(new myErrorHandler());
domDoc = builder.parse(fileName);
```
- Parser properties can be configured:
 - for both SAXParserFactories and DocumentBuilderFactorys (before parser/builder creation):


```
factory.setValidating(true/false)
factory.setNamespaceAware(true/false)
```

SDPL 2008

3.3: (XML APIs) JAXP

14

JAXP: Controlling parsing (2)

- Further DocumentBuilderFactory configuration methods to control the form of the resulting DOM Document:

```
dbf.setIgnoringComments(true/false)
dbf.setIgnoringElementContentWhitespace(true/false)
dbf.setCoalescing(true/false)
    • combine CDATA sections with surrounding text?
dbf.setExpandEntityReferences(true/false)
```

SDPL 2008

3.3: (XML APIs) JAXP

15

DOM vs. Other Java/XML APIs

- JDOM (www.jdom.org), DOM4J (www.dom4j.org), JAXB (java.sun.com/xml/jaxb)
- The others may be more convenient to use, but ...
 - “The **DOM** offers not only the ability to move between languages with minimal relearning, but to move between multiple implementations in a single language – which a specific set of classes such as JDOM can't support”
 - » J. Kesselman, IBM & W3C DOM WG

SDPL 2008

3.3: (XML APIs) JAXP

16

JAXP Transformation API

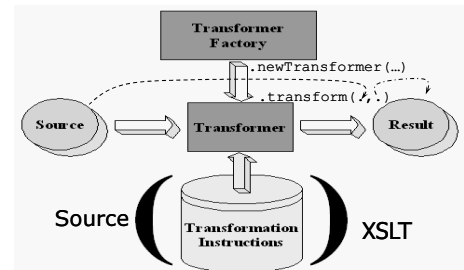
- Package **`javax.xml.transform`**
 - `TransformerFactory` and `Transformer` classes; initialization similar to parser factories and parsers
- Allows application to apply a `Transformer` to a `Source` document to get a `Result` document
- `Transformer` can be created
 - from an XSLT script
 - without instructions → an identity transformation from a `Source` to the `Result`

SDPL 2008

3.3: (XML APIs) JAXP

17

JAXP: Using Transformers (1)



SDPL 2008

3.3: (XML APIs) JAXP

18

Transformation Source & Result

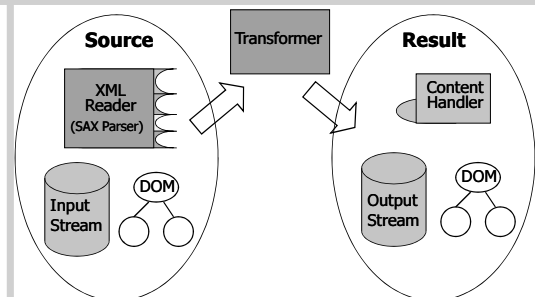
- Transformation `Source` object can be
 - (a node of) a DOM tree
 - a SAX `XMLReader` or
 - an input stream
- Transformation `Result` object can be
 - (a node of) a DOM tree
 - a SAX `ContentHandler` or
 - an output stream

SDPL 2008

3.3: (XML APIs) JAXP

19

Source-Result combinations



SDPL 2008

3.3: (XML APIs) JAXP

20

JAXP Transformation Packages

- Classes to create `Source` and `Result` objects from DOM, SAX and I/O streams defined in packages
 - `javax.xml.transform.dom`,
 - `javax.xml.transform.sax`, and
 - `javax.xml.transform.stream`
- Identity transformation to an output stream is a vendor-neutral way to serialize DOM documents
 - as an alternative to DOM3 Save

SDPL 2008

3.3: (XML APIs) JAXP

21

Serializing a DOM Document as XML text

- By an identity transformation to an output stream:
- ```

 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);

```

SDPL 2008

3.3: (XML APIs) JAXP

22

## Controlling the form of the result?

- We could specify the requested form of the result by an XSLT script, say, in file `saveSpec.xslt`:

```

<xsl:transform version="1.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 <xsl:output encoding="ISO-8859-1" indent="yes"
 doctype-system="regist.dtd" />

 <xsl:template match="/">
 <!-- copy whole document: -->
 <xsl:copy-of select="." />
 </xsl:template>
</xsl:transform>

```

SDPL 2008

3.3: (XML APIs) JAXP

23

## Creating an XSLT Transformer

- Create a tailored transformer:
- ```

StreamSource saveSpecSrc =
    new StreamSource(
        new File("saveSpec.xslt"));
Transformer transformer =
    tFactory.newTransformer(saveSpecSrc);
// and use it to transform a Source to a Result,
// as before
    
```
- The `Source` of transformation instructions could be given also as a `DOMSource` or `SAXSource`

SDPL 2008

3.3: (XML APIs) JAXP

24

Transformation OutputProperties

```
Transformer myTr = tFactory.newTransformer();
// Set identity transformer's output properties:
myTr.setOutputProperty(OutputKeys.ENCODING,
    "iso-8859-1");
myTr.setOutputProperty(OutputKeys.DOCTYPE_SYSTEM,
    "regist.dtd");
myTr.setOutputProperty(OutputKeys.INDENT, "yes");
// Then use it as before
```

- Equivalent to the previous "saveSpec.xslt" Transformer

SDPL 2008

3.3: (XML APIs) JAXP

25

Stylesheet Parameters

- Can also pass parameters to a transformer created from a script like this:

```
<xsl:transform ... >
  <xsl:output method="text" />
  <xsl:param name="In" select="0" />
  <xsl:template match="/">
    <xsl:value-of select="2*$In"/>
  </xsl:template>
</xsl:transform>
```

default value

using

```
myTrans.setParameter("In", 10)
```

SDPL 2007

XSLT: Additional Features and Computing

26

JAXP Validation

- JAXP 1.3 introduced also a Validation framework
 - based on familiar Factory pattern, to provide independence of schema language and implementation
 - » SchemaFactory → Schema → Validator
 - separates validation from parsing
 - » say, to validate an in-memory DOM subtree

SDPL 2008

3.3: (XML APIs) JAXP

27

Validation Example: "Xeditor"

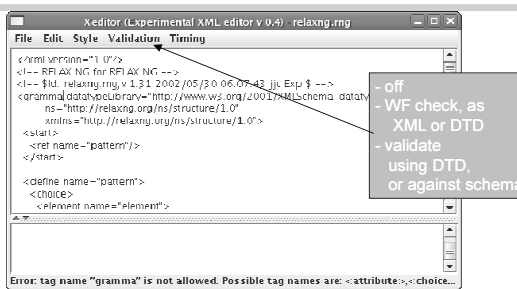
- Xeditor, an experimental XML editor
 - to experiment and demonstrate JAXP-based on-the-fly multi-schema validation
 - M. Saesmaa and P. Kilpeläinen: On-the-fly Validation of XML Markup Languages using off-the-shelf Tools. Extreme Markup Languages 2007, Montréal, August 2007

SDPL 2008

3.3: (XML APIs) JAXP

28

Look & Feel of "Xeditor"



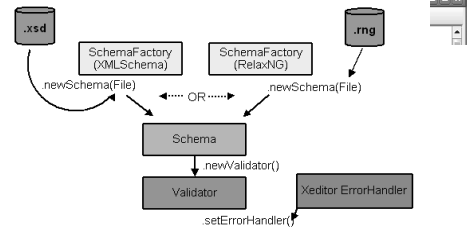
SDPL 2008

3.3: (XML APIs) JAXP

29

Different Schemas and Schema Languages

- A Validator created when the user selects Schema



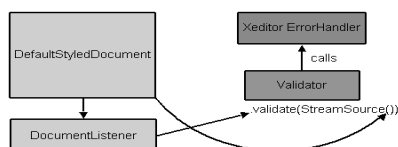
EML' 07 Montreal

On-the-fly Validation of XML

30

Event-driven document validation

- Modified document passed to the Validator
 - errors caught as SAX parse exceptions



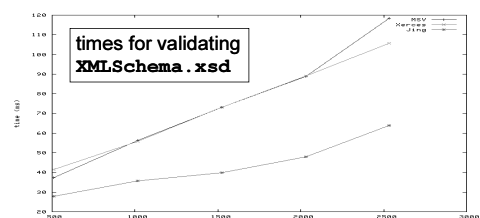
EML' 07 Montreal

On-the-fly Validation of XML

31

Efficiency of In-Memory Validation

- Is brute-force re-validation too inefficient?
- No: Delays normally unnoticeable



EML' 07 Montreal

On-the-fly Validation of XML

32

JAXP: Summary

- An interface for using XML Processors
 - SAX/DOM parsers, XSLT transformers
 - schema-based validators (in JAXP 1.3)
- Supports pluggability of XML processors
- Defines means to control parsing, and handling of parse errors (through SAX ErrorHandlers)
- Defines means to create and save DOM Documents