XML: The Evolution of JAXP

Rahul Srivastava

Project Lead
Oracle

http://www.oracle.com

TS-4743

Understanding XML and Java™ API for XML Processing (JAXP)

Leverage the Best Out of JAXP 1.3

Learn what's new in JAXP 1.3 and how you can use them to process your XML documents in a better way.

Agenda

JAXP Overview

XML and Unicode

XML Parsing

XML Validation

XPath Evaluation

XML Transformation

JAXP Pluggability Layer

Much Better JAXP

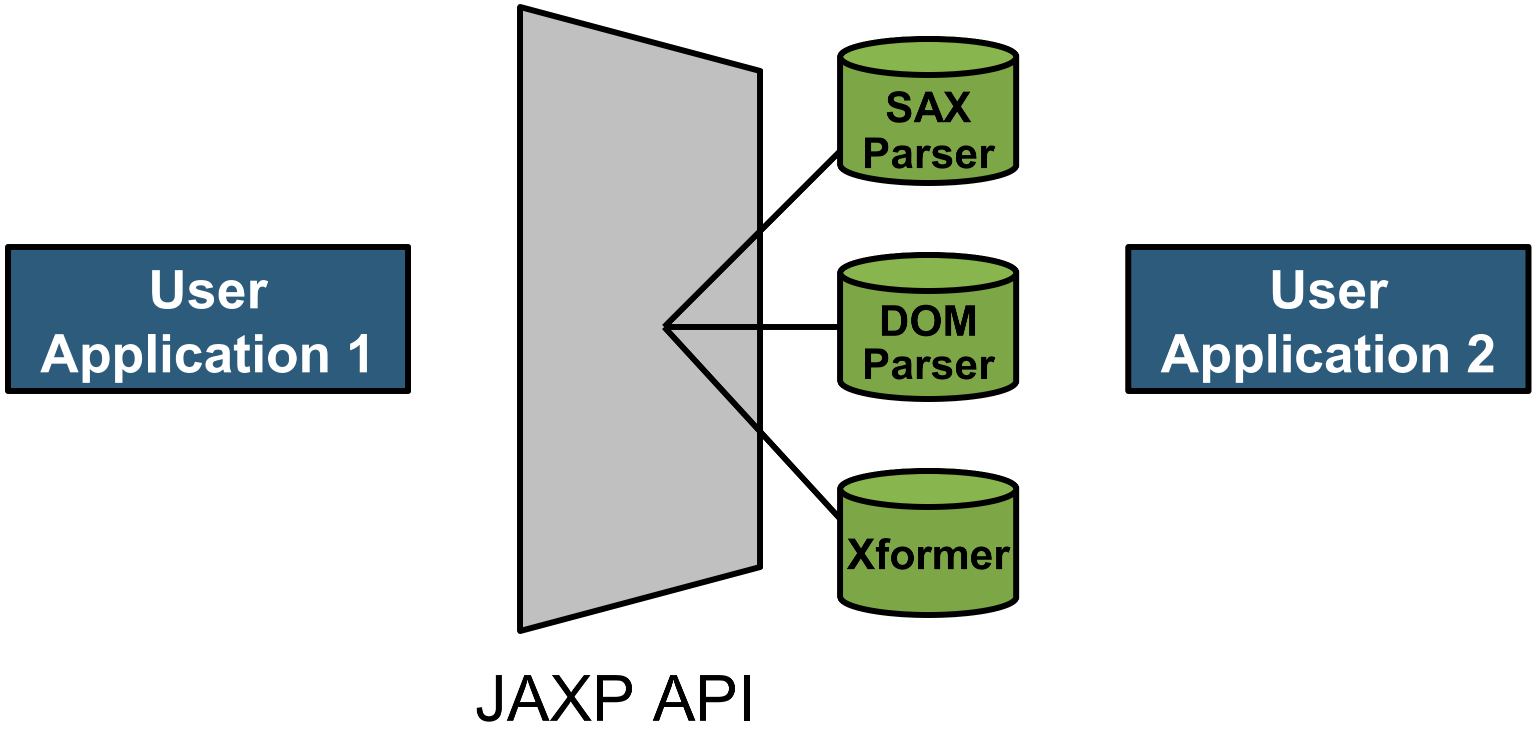
JAXP Overview

What Is It?

- A lightweight and pluggable API for processing XML documents
- JAXP supports
  - XML parsing using SAX and DOM
  - XML instance validation
    - DTD, XMLSchema, and other grammars like RELAX NG
    - While parsing and without parsing
  - XPath evaluation
  - XML transformation using XSL-T

JAXP Overview (Cont.)

Application View of JAXP



JAXP Overview (Cont.)

What's New in JAXP 1.3?

- XML 1.1 and Namespaces in XML 1.1
- XML Inclusions—XInclude 1.0
- Validation of instance against pre-parsed schema
- Evaluating XPath expressions
- XML/Java type mappings for data types defined in XMLSchema 1.0, XQuery 1.0 and XPath 2.0 data model
- DOM L3 and SAX 2.0.2
- Feature for secure processing of XML documents

Agenda

JAXP Overview

XML and Unicode

XML Parsing

XML Validation

XPath Evaluation

XML Transformation

JAXP Pluggability Layer

Much Better JAXP

XML and Unicode

Encoding Is Very Important

- XML inherently supports Unicode
  - Unicode characters can be used in the names of elements, character data, names of attributes, and in the attribute values
- XML 1.0 is backward compatible with Unicode
- XML 1.1 is backward as well as forward compatible with Unicode
- Sample XML

```
<?xml version="1.0" encoding="UTF-8" ?>
< 日本語 で =" ラフル "> こんにちは世界 </ 日本語 >
```

Agenda

JAXP Overview

XML and Unicode

XML Parsing

XML Validation

XPath Evaluation

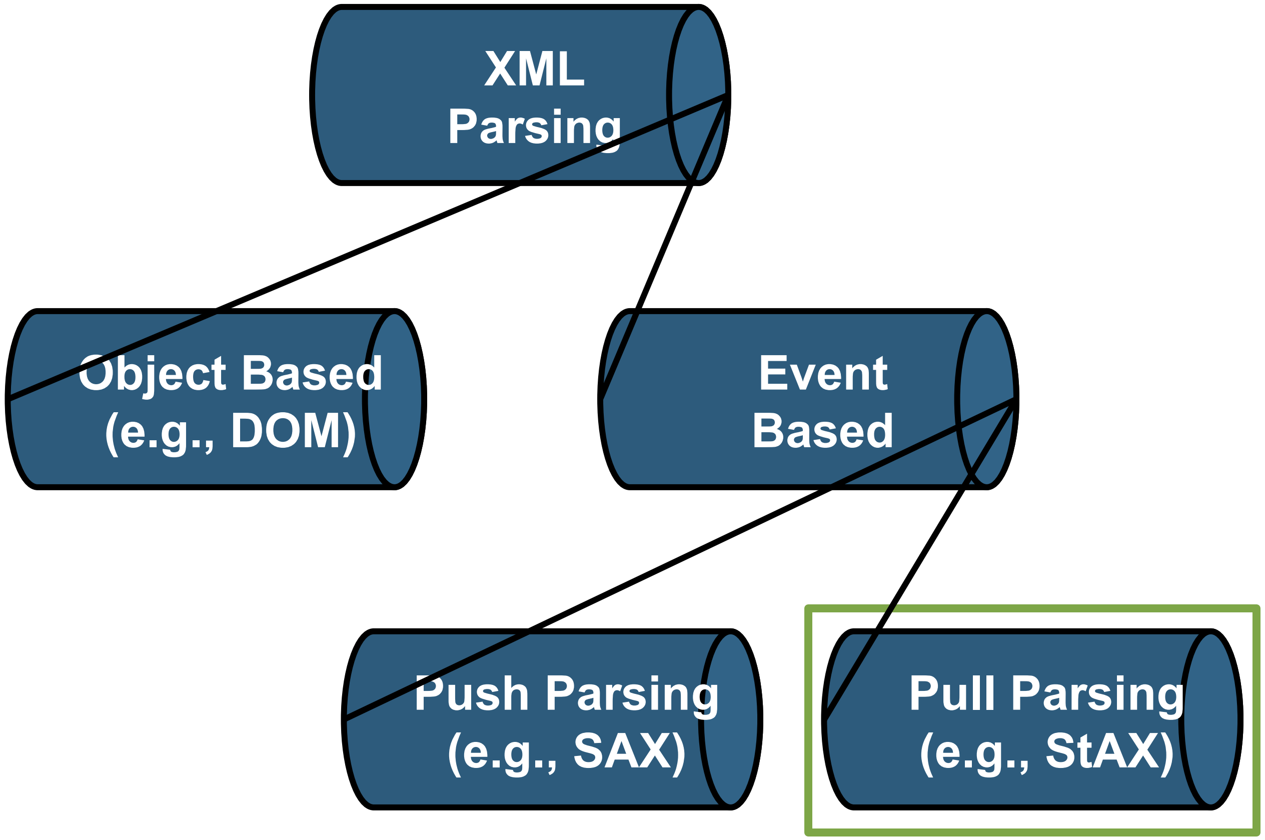
XML Transformation

JAXP Pluggability Layer

Much Better JAXP

XML Parsing

General Classification



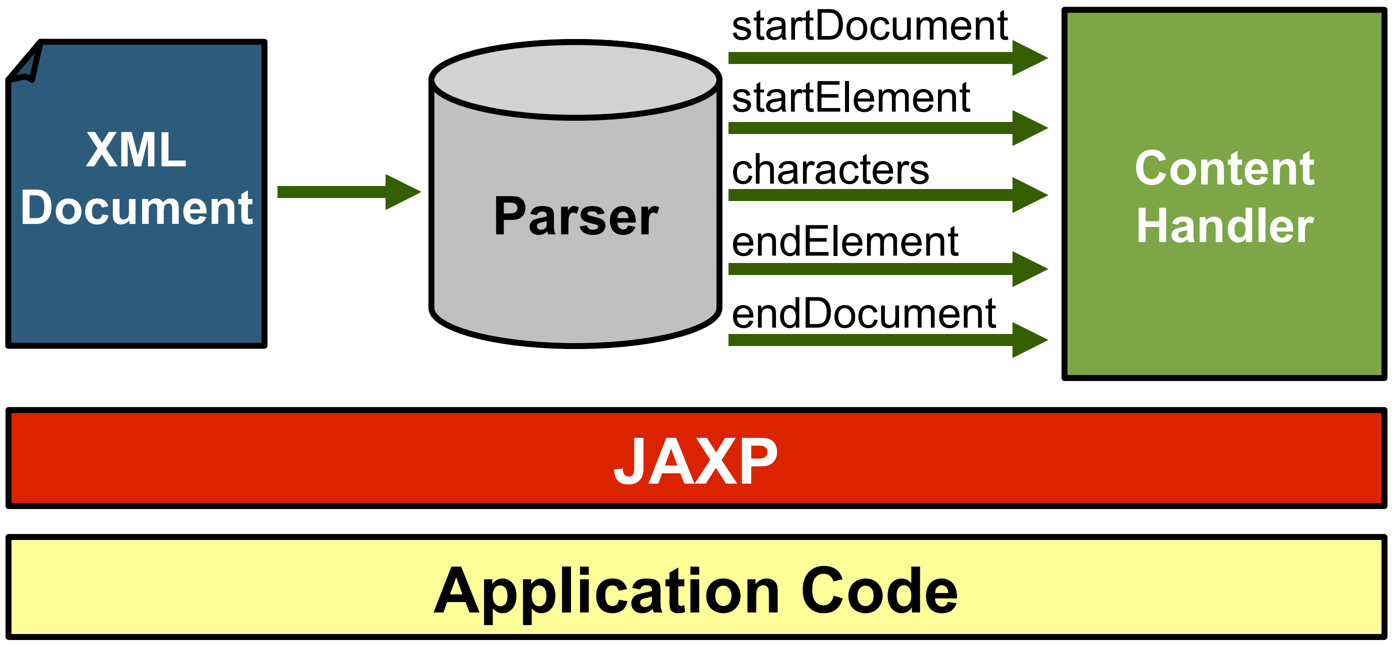
XML Parsing (Cont.)

SAX Parsing

- **S**imple **A**PI for **X**ML (since 1998)
- De facto industry standard
- Parses document sequentially
- Fast and lightweight
- Harder to program!?
- Packaged in:
  - org.xml.sax.*
  - org.xml.sax.ext.*
  - org.xml.sax.helpers.*

XML Parsing (Cont.)

Application View of SAX Parsing



SAX ContentHandler

```
public class MyHandler implements ContentHandler {

  public void startElement(...) throws SAXException {
      //Receive notification for start of an element tag
  }

  public void endElement(...) throws SAXException {
      //Receive notification for end of an element tag
  }

  public void characters(...) throws SAXException {
      //Receive notification for character data.
      //Remember - This method can be invoked multiple
      //times by the parser.
  }
...
}
```

SAX ErrorHandler

```
public class MyErrorHandler implements ErrorHandler {

  public void warning(SAXParseException ex)
              throws SAXException {
    System.out.println("[WARNING] "+ex.getMessage());
  }

  public void error(SAXParseException ex)
              throws SAXException {
    System.out.println("[ERROR] "+ex.getMessage());
  }

  public void fatalError(SAXParseException ex)
              throws SAXException {
    System.out.println("[FATAL] "+ex.getMessage());
  }

}
```

SAX EntityResolver

```
public class MyEntityRes implements EntityResolver {

  public InputSource resolveEntity(String publicId,
                                   String systemId) {
    if (blah) {
        return new InputSource(baseId + systemId);
    }

    //otherwise use the default identifiers

    return null;
  }

}
```

SAX Parsing Using JAXP

```
//get the factory
SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setNamespaceAware(true);

//create the sax parser
SAXParser parser = factory.newSAXParser();

//create a single handler which acts as the content
//handler, error handler, and the entity resolver
DefaultHandler handler = new MyHandler();

//parse the xml
parser.parse(“file:///home/foo.xml”, handler);
```

Filtering SAX Events

```
//create the XMLReader for parsing xml
SAXParserFactory spf = SAXParserFactory.newInstance();
SAXParser parser = spf.newSAXParser();
XMLReader reader = parser.getXMLReader();

//setup the filter chain
XMLFilter filter1 = new Filter1(reader);
XMLFilter filter2 = new Filter2(filter1);
filter2.setContentHandler(contenthandler);

//start the parsing
filter2.parse(args[0]);

//XMLReader --> Filter1 --> Filter2 --> ContentHandler
```

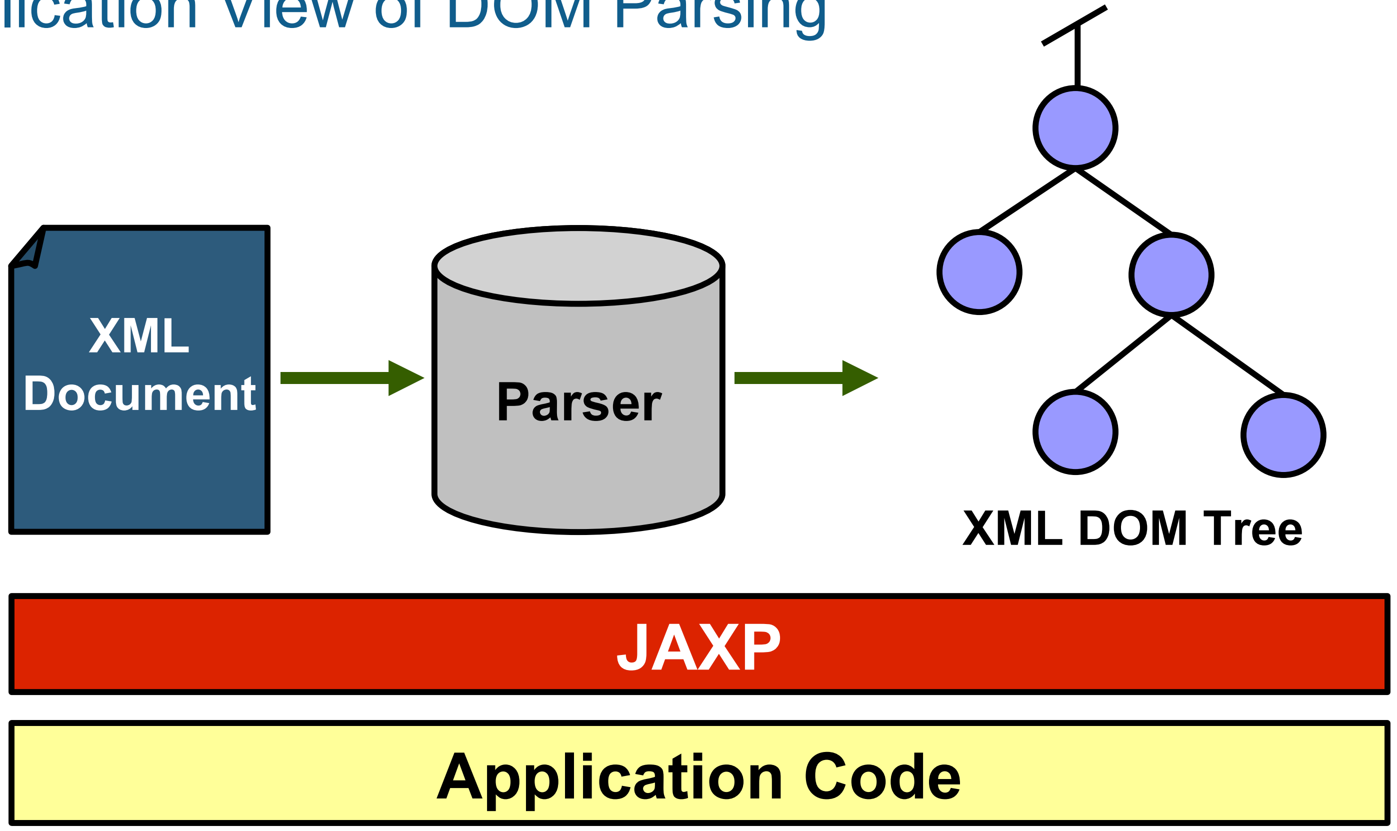
XML Parsing (Cont.)

DOM Parsing

- **D**ocument **O**bject **M**odel (since 1998)
- W3C standard to access XML document via a tree structure, which can be walked back and forth
- Composed of nodes, e.g., element, and text nodes
- Nodes can be added, deleted, modified
- Larger memory requirements
- Allows to create the entire tree from scratch, in-memory
- Packaged in:
    - org.w3c.dom.*

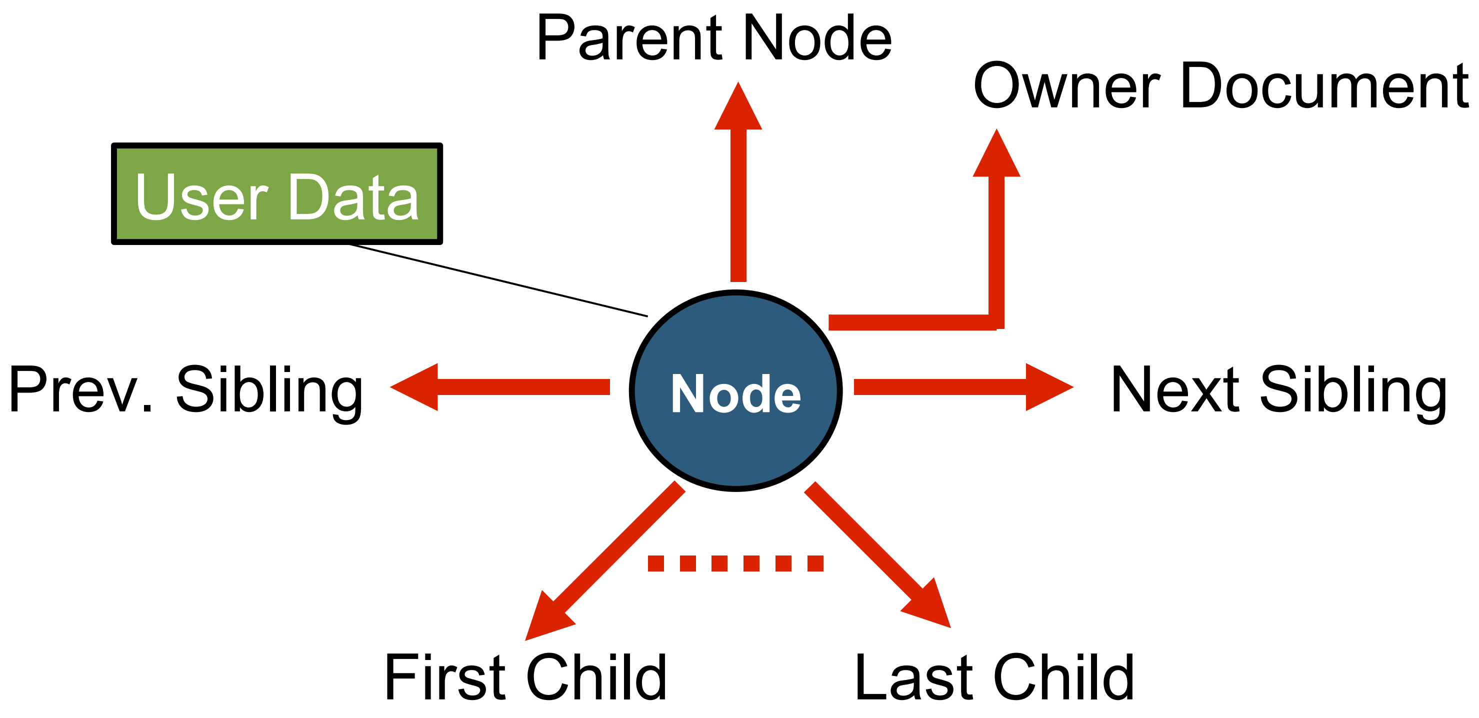
XML Parsing (Cont.)

Application View of DOM Parsing



XML Parsing (Cont.)

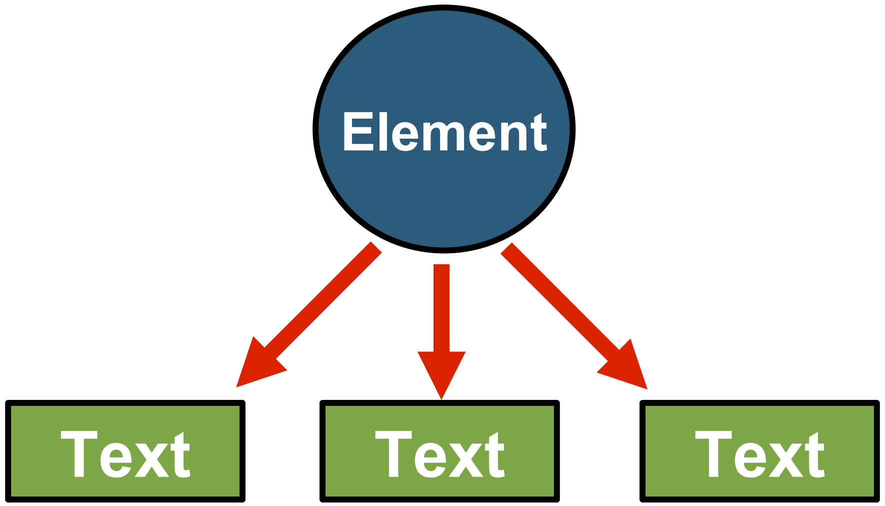
Node of a DOM Tree



XML Parsing (Cont.)

Relation Between Element and Text Node

- Text nodes are the child nodes of an element node, and are always the leaf nodes in the DOM tree



DOM Parsing Using JAXP

```
//get the factory
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
factory.setValidating(true);

//create the dom parser
DocumentBuilder builder = factory.newDocumentBuilder();

//parse the xml document
Document doc = builder.parse(“foo.xml”);

//now you can traverse the DOM tree returned
//using the standard org.w3c.dom APIs
```

Traversing DOM Using W3C APIs

```
//Get the root element from the document node
Element rootElement = doc.getDocumentElement();

//Get the first child of the root element
Node node = rootElement.getFirstChild();

//Get all the attributes for this node
NamedNodeMap attrs = node.getAttributes();

//Get all nodes which have the tag name foo
NodeList list = doc.getElementsByTagName("foo");

...
```

Agenda

JAXP Overview

XML and Unicode

XML Parsing

XML Validation

XPath Evaluation

XML Transformation

JAXP Pluggability Layer

Much Better JAXP

XML Validation

What Does JAXP Support?

- JAXP supports xml instance validation
  - While parsing xml, or
  - Against pre-parsed schema
- The grammar for the instance can be:
  - DTD
  - XMLSchema
  - RELAX NG
  - Or anything else

XML Validation (Cont.)

Validation Against DTD

- An XML instance referencing a DTD, against which this instance would be validated

```
<?xml version="1.0">
<!DOCTYPE root SYSTEM "MyDTD.dtd">
<root>
...
</root>
```

XML Validation Against DTD Using JAXP

```
DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();
OR
SAXParserFactory factory = SAXParserFactory.newInstance();

//this will validate against DTD
factory.setValidating(true);

//If the validation is turned off,
//would the referenced DTD be loaded?
```

XML Validation (Cont.)

Validation Against XMLSchema

- An XML instance document referencing an XMLSchema document, against which this instance would be validated

```
<?xml version="1.0"?>
<root
     xmlns:xsi=“http://www.w3.org/2001/XMLSchema-instance”
     xsi:schemaLocation=“foo MyXSD.xsd”
     xmlns=“foo”>
...
</root>
```

XML Validation Against XMLSchema Using JAXP

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

//this will validate against referenced XMLSchema
factory.setNamespaceAware(true);
factory.setValidating(true);
factory.setAttribute(
 "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
 "http://www.w3.org/2001/XMLSchema");

...

//If the XML instance has a reference to both the DTD and
//XMLSchema, then in that case, against what grammar the
//instance would be validated?
```

XML Validation (Cont.)

Validation Against…

- An XML instance file not referencing any grammar; Can we validate this?

```
<?xml version="1.0"?>
<root xmlns="foo">
...
</root>
```

Validating Against Externally Supplied XMLSchema

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

factory.setNamespaceAware(true);
factory.setValidating(true);

//this will validate against externally supplied XMLSchema
factory.setAttribute(
 "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
 "http://www.w3.org/2001/XMLSchema");

factory.setAttribute(
 "http://java.sun.com/xml/jaxp/properties/schemaSource",
 "file:///home/xsd/foo.xsd");

...
```

Validating Against Externally Supplied RELAX NG

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

factory.setNamespaceAware(true);
factory.setValidating(true);

//this will validate against externally supplied XMLSchema
factory.setAttribute(
 "http://java.sun.com/xml/jaxp/properties/schemaLanguage",
 "http://relaxng.org/ns/structure/1.0");

factory.setAttribute(
 "http://java.sun.com/xml/jaxp/properties/schemaSource",
 "file:///home/xsd/foo.rng");

...
```

Validating Against Pre-parsed Schema

```
//create a SchemaFactory for loading W3C XML Schemas
SchemaFactory wxsfactory =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_U
RI);

//set the errorhandler for errors in schema itself
wxsfactory.setErrorHandler(schemaErrorHandler);

//load the W3C XMLSchema
Schema schema = wxsfactory.newSchema(new File(args[0]));

//create a validator from the loaded schema
Validator validator = schema.newValidator();

//set the errorhandler for validation errors
validator.setErrorHandler(validationErrorHandler);

//validate the XML instance
validator.validate(xmlsource);
```

Validating Against Pre-parsed Schema While Parsing XML

```
//create a SchemaFactory for loading W3C XML Schemas
SchemaFactory wxsfactory =
SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_U
RI);

//set the errorhandler for errors in schema itself
wxsfactory.setErrorHandler(schemaErrorHandler);

//load the W3C XMLSchema
Schema schema = wxsfactory.newSchema(new File(args[0]));

//create the parser factory
SAXParserFactory spfactory =
SAXParserFactory.newInstance();

//set the pre-parsed schema
Spfactory.setSchema(schema);
...
```

Agenda

JAXP Overview

XML and Unicode

XML Parsing

XML Validation

XPath Evaluation

XML Transformation

JAXP Pluggability Layer

Much Better JAXP

XPath
What Is It?

- It’s a language to address parts of an XML document
- It uses UNIX-like expression
  - For example: /home/rahsriva/
- The result of an XPath expression can be:
  - Set of nodes (aka “node-set”)
  - Boolean
  - Number
  - String (Unicode characters)

XPath (Cont.)

Some More Details

- An XPath expression is made up of Location Paths, and XPath functions
- Each Location Path is made up of Steps separated by “/”
- Each Step is made up of an Axis, and a Node test; and the Step can further be refined using Predicates
- For example: /foo/bar[@baz]

Evaluating XPath Expressions Using JAXP

```
//get the XPath processor
XPathFactory xpfactory = XPathFactory.newInstance();
XPath xpathprocessor = xpfactory.newXPath();

//create an XPath expression
XPathExpression employeesXPath =
xpathprocessor.compile("/employees/employee");

//execute the XPath expressions
NodeList employees =
(NodeList)employeesXPath.evaluate(doc,
XPathConstants.NODESET);

//print the result
for (int i=0; i<employees.getLength(); i++) {
  System.out.println(employees.item(i).getTextContent());
}
```

Agenda

JAXP Overview

XML and Unicode

XML Parsing

XML Validation

XPath Evaluation

XML Transformation

JAXP Pluggability Layer

Much Better JAXP

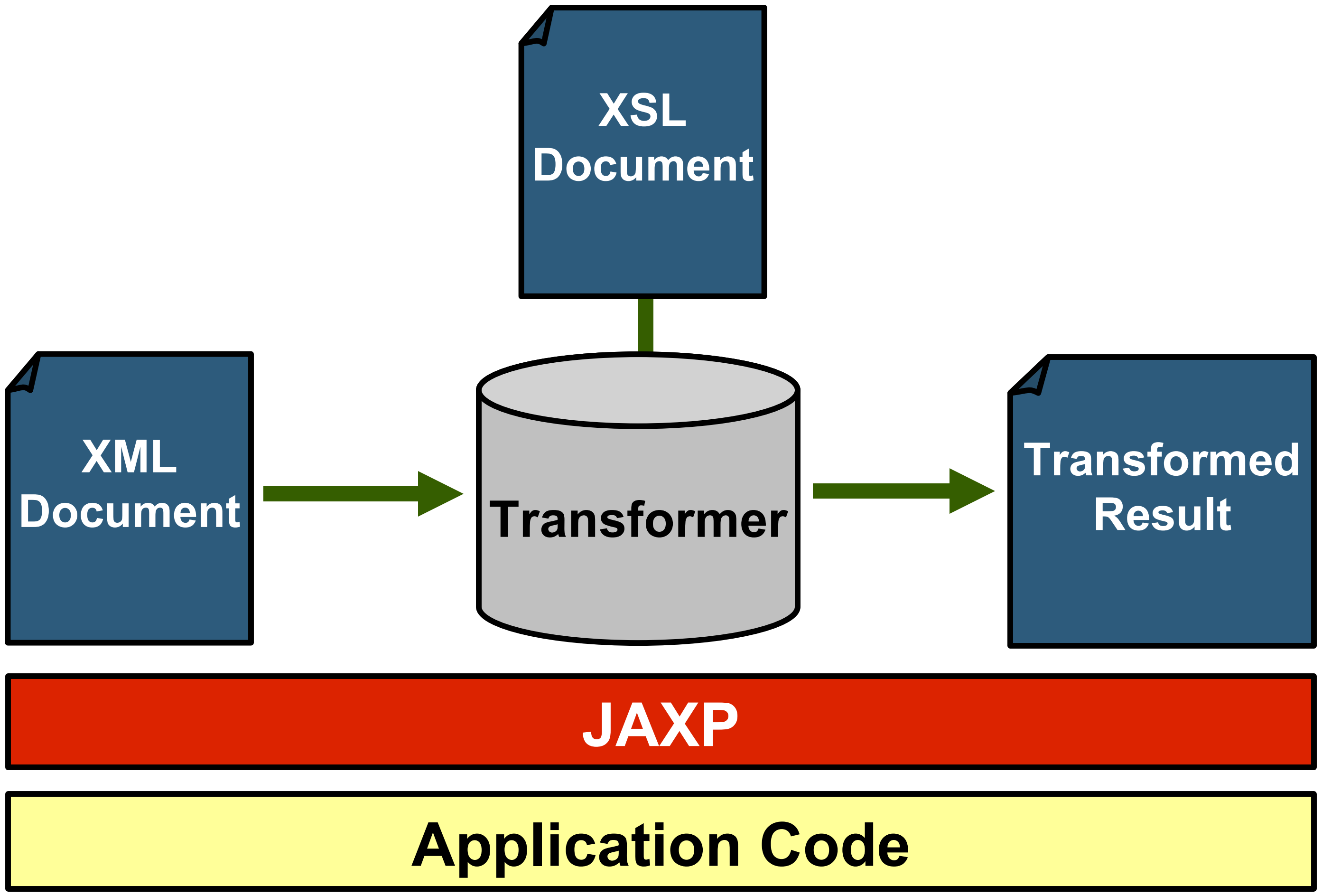
XML Transformation

XSL-T and TrAX

- XSL-T is a syntax and semantics for transforming XML documents into other XML documents or any other format
- W3C XSL-T does not define APIs for transformation
- JAXP defines extensive set of XSLT APIs
- This is what used to be known as TrAX
- TrAX was rolled into JAXP (since JSR-63) to have better support for XSLT in the Java platform

XML Transformation (Cont.)

Application View of Transformation



XML Transformation (Cont.)

javax.xml.transform

- Defines basic set of interfaces for XSLT processors
- Defines TransformerFactory and Transformer abstract classes that all processors implement
- Defines Templates, Source, and Result interfaces
- Templates represent processed transformation instructions

XML Transformation (Cont.)

Source and Result

- Specialized implementations for Source and Result available in:
  - javax.xml.transform.dom
  - javax.xml.transform.sax
  - javax.xml.transform.stream
- Different combinations of Source and Result can be passed to the Transformer class

XML Transformation Using JAXP

```
TransformerFactory factory =
  TransformerFactory.newInstance();

//Create a transformer using a particular stylesheet
Transformer transformer =
factory.newTransformer(new StreamSource(“foo.xsl”));

//Transform the source xml to result using the above XSL
transformer.transform(new StreamSource(“foo.xml”),
                      new StreamResult(System.out));
```

Serializing a DOM

```
Document doc;
...
TransformerFactory tfactory =
  TransformerFactory.newInstance();

//create a transformer without using any XSL
Transformer serializer = tfactory.newTransformer();

serializer.transform(new DOMSource(doc),
                     new StreamResult(System.out));

//Replace StreamResult with SAXResult in the above example
//and it would generate SAX events from the given DOM
```

Agenda

JAXP Overview

XML and Unicode

XML Parsing

XML Validation

XPath Evaluation

XML Transformation

JAXP Pluggability Layer

Much Better JAXP

JAXP Pluggability Layer

Plugging Parsers, Transformers, etc.

- Factory lookup is accomplished by:
  - System property
    - javax.xml.xxx.yyyFactory
    - For example: javax.xml.parsers.DocumentBuilderFactory
  - $JAVA_HOME/lib/jaxp.properties file
  - Jar Services API
    - META-INF/services/javax.xml.parsers.XXXFactory
  - Reference Default
- Note: The lookup is done in the above order

JAXP Pluggability Layer (Cont.)
What Can be Plugged?

- Where javax.xml.xxx.yyyFactory can be one of the following:
  - javax.xml.parsers.SAXParserFactory
  - javax.xml.parsers.DocumentBuilderFactory
  - javax.xml.transform.TransformerFactory
  - javax.xml.xpath.XPathFactory
  - javax.xml.validation.SchemaFactory:schemaLanguage
    - schemaLanguage is the parameter passed to the newInstance method of SchemaFactory

Agenda

JAXP Overview

XML and Unicode

XML Parsing

XML Validation

XPath Evaluation

XML Transformation

JAXP Pluggability Layer

Much Better JAXP

Much Better JAXP

Things That Would Make JAXP Even Better

- If StAX is made as part of JAXP
- If event-based transformation is supported
- If there are APIs available to use W3C XMLSchema datatypes
- If there are APIs available to traverse the abstract model of XMLSchema

Speak Out
Participate in the Process

Send comments, feedback to:

- Jeff.Suttor@Sun.com (JAXP Spec Lead)
- JSR-206-comments@JCP.org

Summary

- The Java API for XML Processing (JAXP) allows you to:
  - Parse XML documents using SAX and DOM
  - Validate an instance document against various schemas
    - While parsing XML instance, or
    - Against pre-parsed schemas
  - Evaluate XPath expressions against an XML document
  - Do transformations using XSLT
  - Plug different parsing, transformation engines, etc. without the need to change a single line of application code

For More Information

http://java.sun.com/webservices/jaxp/

Q&A

XML: The Evolution of JAXP

Rahul Srivastava

Project Lead
Oracle

http://www.oracle.com

TS-4743