JavaOne℠

Sun's 2002 Worldwide Java Developer Conference℠

JDOM Makes XML Easy

Jason Hunter
Co-Creator
JDOM Project

JDOM Makes XML Easy

"I think JDOM breaks down a lot of the barriers between Java and XML and makes it a more natural fit."

-- Simon St. Laurent,
Author *XML Elements of Style*

What is JDOM?

JDOM is an open source library for Java-optimized XML data manipulations

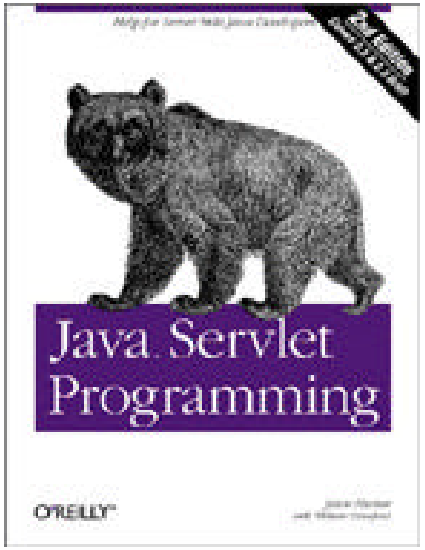
This session provides a technical introduction to JDOM

Learning Objectives

- This presentation will teach you...
  - Why (and how) we created JDOM
  - Where JDOM is most useful
  - How to program with JDOM
  - How JDOM compares to other technologies
  - What's new with Beta 8
  - What life is like as an open source JSR

Introductions

- Jason Hunter created JDOM (along with Brett McLaughlin)
  - Jason works as an independent consultant
  - Lead author of *Java Servlet Programming, 2nd Edition* (O'Reilly, http://www.servlets.com)



Agenda

- The JDOM Philosophy
- The JDOM Classes
- Current Code Status
- JDOM as a JSR

The JDOM Project

- JDOM is...
  - A programming model to represent XML data
  - Similar to the DOM but not built on DOM or modeled after DOM
  - An open source project with an Apache license and 2000+ list subscribers
  - A Java Specification Request (JSR-102)

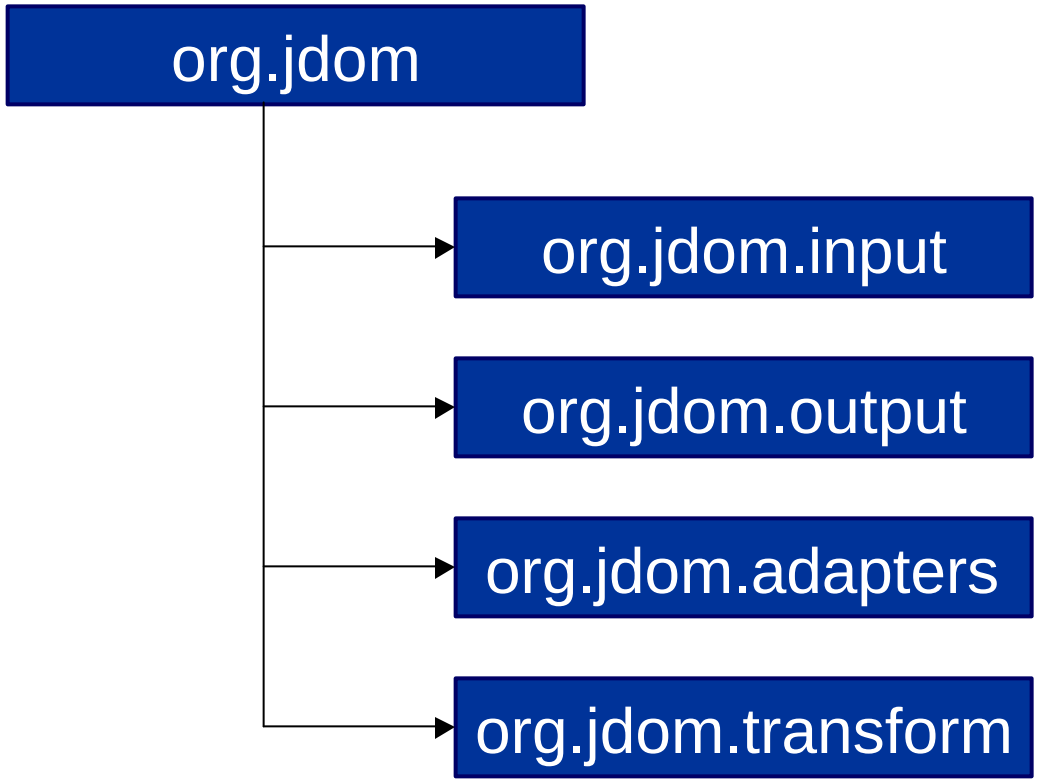
The JDOM Philosophy

- JDOM was created to...
  - Be straightforward for Java programmers
  - Use the power of the Java language (method overloading, collections, reflection)
  - Hide the complexities of XML wherever possible
  - Integrate well with SAX and DOM

Scratching an Itch

- JDOM was born in the spring of 2000
  - Because DOM and SAX weren't sufficient
  - SAX has no document modification, random access, or output abilities
  - SAX often requires building a state machine
  - DOM feels foreign to the Java programmer
  - DOM is defined in IDL, a lowest common denominator across languages

Package Structure



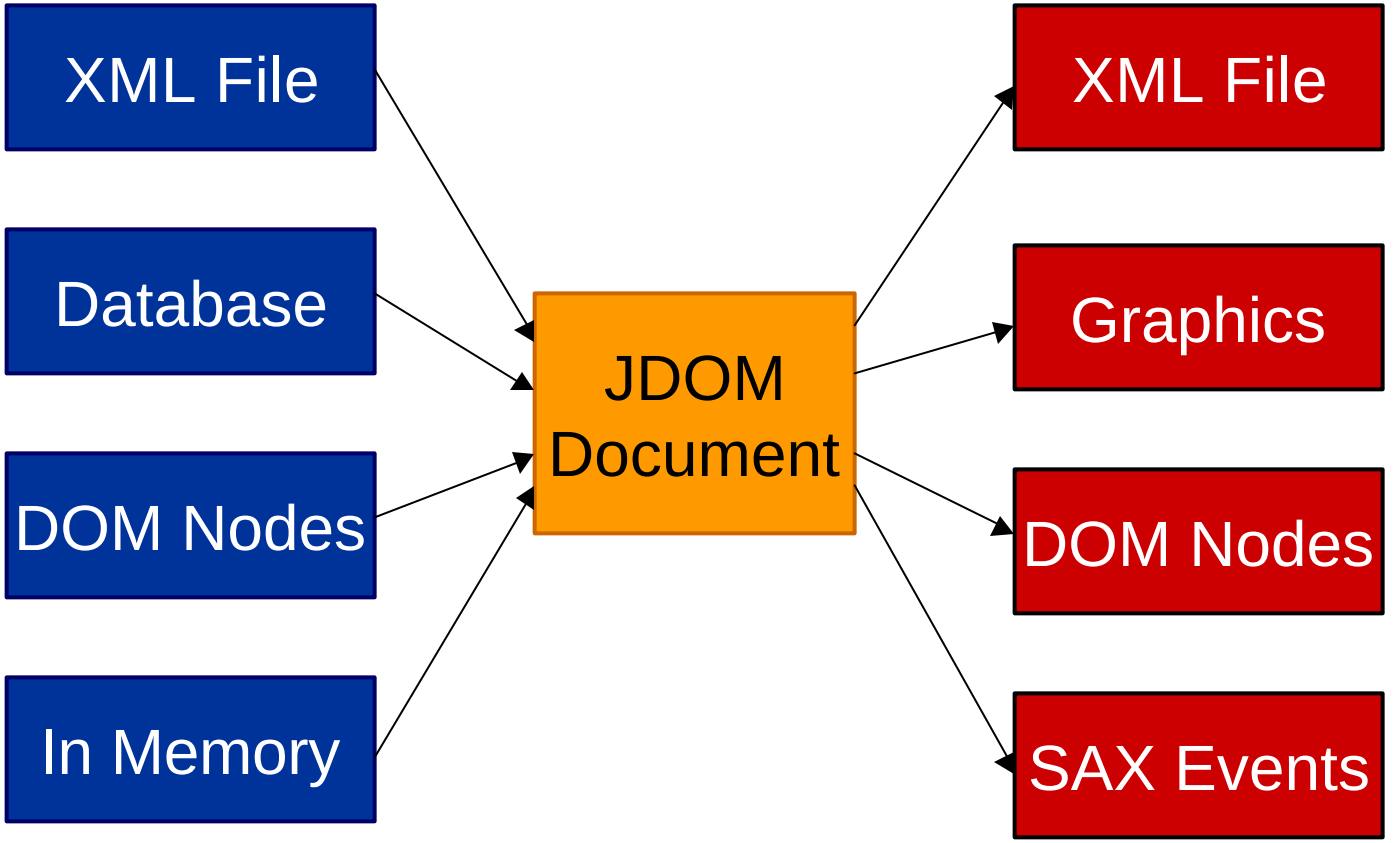
The JDOM Classes

- **The org.jdom Package**
  - Attribute
  - CDATA
  - Comment
  - DocType
  - Document
  - Element
  - EntityRef
  - Namespace
  - ProcessingInstruction
  - Text
- **The org.jdom.transform Package**
  - JDOMSource
  - JDOMResult

The JDOM Classes

- **The org.jdom.input Package**
  - SAXBuilder
  - DOMBuilder
  - *ResultSetBuilder*
- **The org.jdom.output Package**
  - XMLOutputter
  - SAXOutputter
  - DOMOutputter
  - *JTreeOutputter*

General Program Flow



JAXP and TRaX

- JDOM supports JAXP™ 1.1
  - Builders can use any parser, but JAXP parsers are the default
  - JAXP.next may support JDOM directly
- TRaX/XSLT transformations in JAXP
  - With **`JDOMSource`** and **`JDOMResult`** classes

The Document Class

- Documents are represented by **`org.jdom.Document`**
- They may be constructed from scratch:

```
Document doc =
  new Document(new Element("root"));
```

- Or built from a file, stream, system ID, URL:

```
SAXBuilder builder = new SAXBuilder();
Document doc = builder.build(url);
```

JDOM vs DOM

- To create a simple document in JDOM:

```
Document doc = new Document();
Element e = new Element("root");
e.setText("This is the root");
doc.addContent(e);
```

- Or for power users:

```
Document doc = new Document(
  new Element("root").setText(
  "This is the root"));
```

JDOM vs DOM

- The same task in JAXP/DOM requires more complex code:

```
DocumentBuilderFactory factory =
  DocumentBuilderFactory.newInstance();
DocumentBuilder builder =
  factory.newDocumentBuilder();
Document doc = builder.newDocument();
Element root =
  doc.createElement("root");
Text text = doc.createText(
  "This is the root");
root.appendChild(text);
doc.appendChild(root);
```

The XMLOutputter Class

- Output is very flexible

```
Document doc = new Document(...);
XMLOutputter outp = new XMLOutputter();

// Raw output
outp.output(doc, fileOutputStream);

// Compressed output
outp.setTextTrim(true);
outp.output(doc, socket.getOutputStream());

// Pretty output
outp.setIndent("  ");
outp.setNewlines(true);
outp.output(doc, System.out);
```

The Element Class

- Here's how to navigate the Element tree

```
// Get the root element
Element root = doc.getRootElement();

// Get a list of all child elements
List allChildren = root.getChildren();

// Get only elements with a given name
List namedChildren = root.getChildren("name");

// Get the first element with a given name
Element child = root.getChild("name");
```

- That **`List`** is a **`java.util.List`**!

Managing Kids

- The List is live!

```
List allChildren = root.getChildren();

// Remove the fourth child
allChildren.remove(3);

// Remove children named "jack"
allChildren.removeAll(root.getChildren("jack"));
root.removeChildren("jack"); // convenience

// Add a new child
allChildren.add(new Element("jane"));
root.addContent(new Element("jane")); // conv.
allChildren.add(0, new Element("first"));
```

Moving Elements

- Moving elements is easy in JDOM

```
Element movable = new Element("movable");
parent1.addContent(movable);    // place
parent1.removeContent(movable); // remove
parent2.addContent(movable);    // add
```

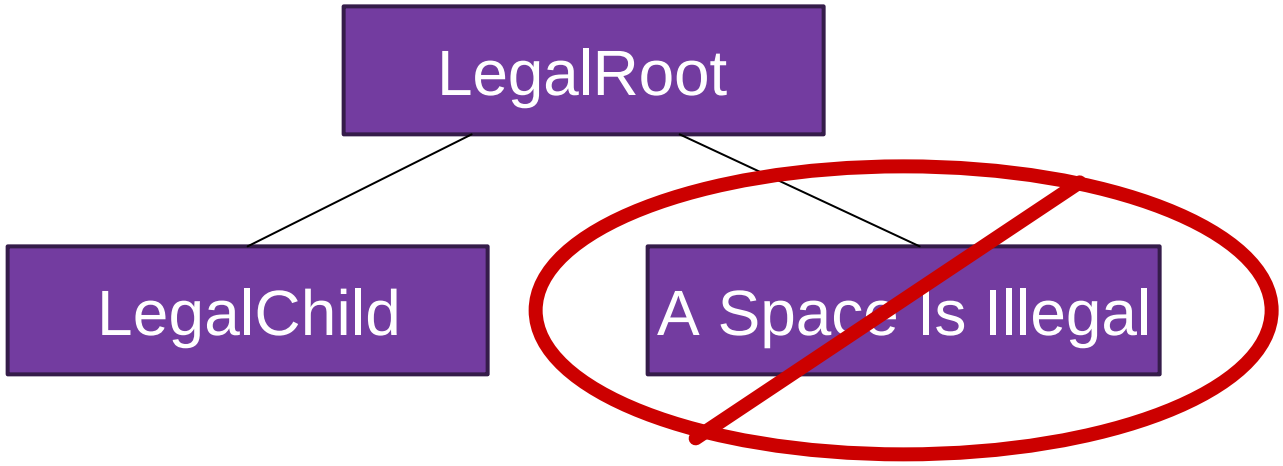
- It's not so easy in DOM

```
Element movable = doc1.createElement("movable");
parent1.appendChild(movable);   // place
parent1.removeChild(movable);   // remove
parent2.appendChild(movable);   // error!
```

- JDOM elements aren't tied to their Document or build tool!

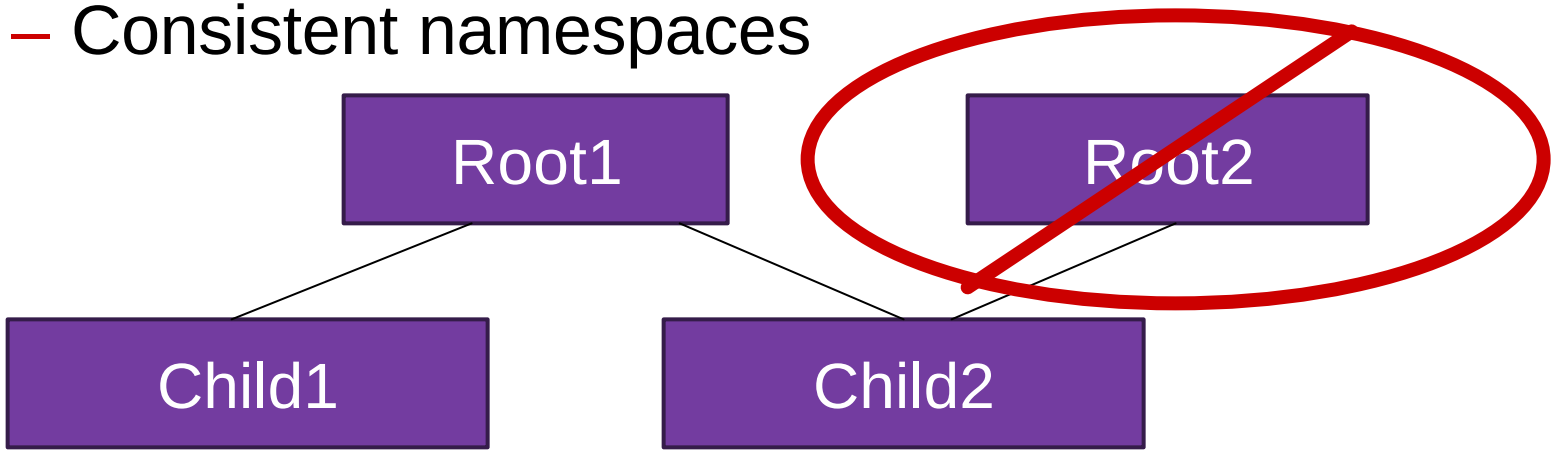
Well-Formedness

- The Element constructor (and others) check the element is legal
  - No inappropriate characters, etc



Well-Formedness

- The add/remove method also check structure
  - No loops in any tree
  - One and only one root element
  - Consistent namespaces



The Attribute Class

- Elements may have attributes

```
<table width="100%" border="0"> </table>
```

```
// Get an attribute
String width = table.getAttributeValue("width");
int border = table.getAttribute("width")
                                  .getIntValue();

// Set an attribute
table.setAttribute("vspace", "0");

// Remove an attribute or all attributes
table.removeAttribute("vspace");
table.getAttributes().clear();
```

Element Content

- An Element may have text content

```
<description>
  A cool demo
</description>
```

```
// The text is directly available
// Returns "\n  A cool demo\n"
String desc = element.getText();

// There's a convenient shortcut
// Returns "A cool demo"
String desc = element.getTextTrim();
```

Element Content

- Text content can be changed directly

```
element.setText("A new description");
```

- Special chars are interpreted correctly

```
element.setText("<xml> content");
```

- You can also create CDATA, but it can be retrieved the standard way

```
element.addContent(new CDATA("<xml> content"));
String noDifference = element.getText();
```

Mixed Content

- Elements may contain many things

```
<table>
  <!-- Some comment -->
  Some text
  <tr>Some child element</tr>
</table>
```

- Standard uses are kept simple!

```
String text = table.getTextTrim();
Element tr = table.getChild("tr");
```

More Mixed Content

- The raw mixed content is also available

```
List mixedCo = table.getContent();
Iterator itr = mixedCo.iterator();
while (itr.hasNext()) {
  Object o = i.next();
  if (o instanceof Comment) {
    ...
  }
  // Also Comment, Element, Text, CDATA,
  // ProcessingInstruction, and EntityRef
}

// Now remove the Comment. Why is it "1"?
mixedCo.remove(1);
```

Processing Instructions

- PIs look like this

```
<?br?>
<?cocoon-process type="xslt"?>
```

- They are easy to work with

```
// Get "target", such as cocoon-process
String target = pi.getTarget();

// Get "data", such as ' type="xslt"'
String data = pi.getData();

// The data is available as attributes!
String type = pi.getValue("type");
```

Namespaces

- JDOM supports namespaces natively

```
<xhtml:html
      xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <xhtml:title>Home Page</xhtml:title>
</xhtml:html>
```

```
Namespace xhtml = Namespace.getNamespace(
  "xhtml", "http://www.w3.org/1999/xhtml");

List kids = html.getChildren("title", xhtml);
Element kid = html.getChild("title", xhtml);

kid.addContent(new Element("table", xhtml));
```

Current Status

- Currently JDOM is in Beta 8
- XPath on top of JDOM is quite far along and usable
  - http://www.jaxen.org
- XSLT support is provided via TRaX
  - And incorporated in **`org.jdom.transform`**
- In-memory validation may squeeze into 1.0

Changes in Beta 8

- Most significant improvements in Beta 8
  - Complete rewrite of the List implementation
  - Added a Text class to help XPath
  - Made I/O classes more featureful
  - Added support for internal DTD subsets
  - Finalized the EntityRef implementation
  - Added Attribute Typing
  - Fleshed out well-formedness checking
  - Fixed bugs, improved performance

Success Story

- Orange (UK Telecom) uses JDOM to support their B2B system "Orange API"
  - A "Where is my nearest" WAP service handles 1,000 requests per minute
  - That's just 1 of 40 services built on XML
  - All their XML processing uses JDOM
  - JDOM supports an XML-RPC / SOAP framework now deployed globally
  - Source: Jools Enticknap, Orange

JDOM as a JSR

- JDOM has been accepted as a Java Specification Request (JSR-102)

- Sun's comment with their YES vote:
- "In general we tend to prefer to avoid adding new APIs to the Java platform which replicate the functionality of existing APIs. However, JDOM does appear to be significantly easier to use than the earlier APIs, so we believe it will be a useful addition to the platform"

What It Means

- What exactly does being a JSR mean?
  - Facilitates JDOM's corporate adoption
  - Opens the door for JDOM to be incorporated into the Java Platform
  - JDOM will still be released as open source
  - Technical discussion will continue to take place on public mailing lists

- `http://java.sun.com/aboutJava/communityprocess/jsr/jsr_102_jdom.html`

The People

- Jason Hunter is the "Specification Lead"
- The initial "Expert Group" in order of acceptance:
  - Brett McLaughlin, Jools Enticknap, James Davidson, Jow Bowbeer, Philip Nelson, Sun Microsystems (Rajiv Mordani)
  - Others are in the process of being added
  - All can contribute to technical discussions

Get Involved!

- You too can get involved
  - http://jdom.org
  - Download the software
  - Read the docs
  - Read the FAQ
  - Sign up for the mailing lists
  - Join the discussion

Summary

- JDOM is an open source library for Java-optimized XML data manipulation
- It was created to be easy to learn, powerful, and natural for Java programmers
- It uses JAXP and can integrate with DOM and SAX
- It's an official JSR with a new Beta 8 release
- It makes XML fun!

Q&A

JavaOne℠

Sun's 2002 Worldwide Java Developer Conference™

BEYOND BOUNDARIES

Session #2105