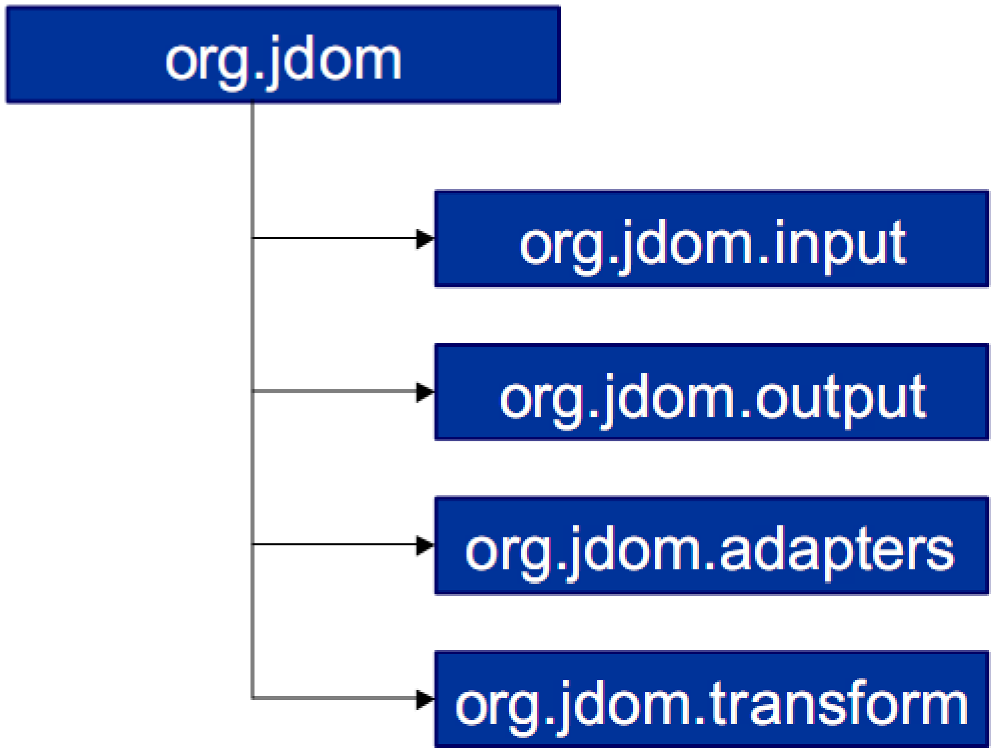
What Is JDOM?

- JDOM is an open source library for Java-optimized XML data manipulations
- 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
- A Java Specification Request (JSR-102)

Rationale

- 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

Package Structure



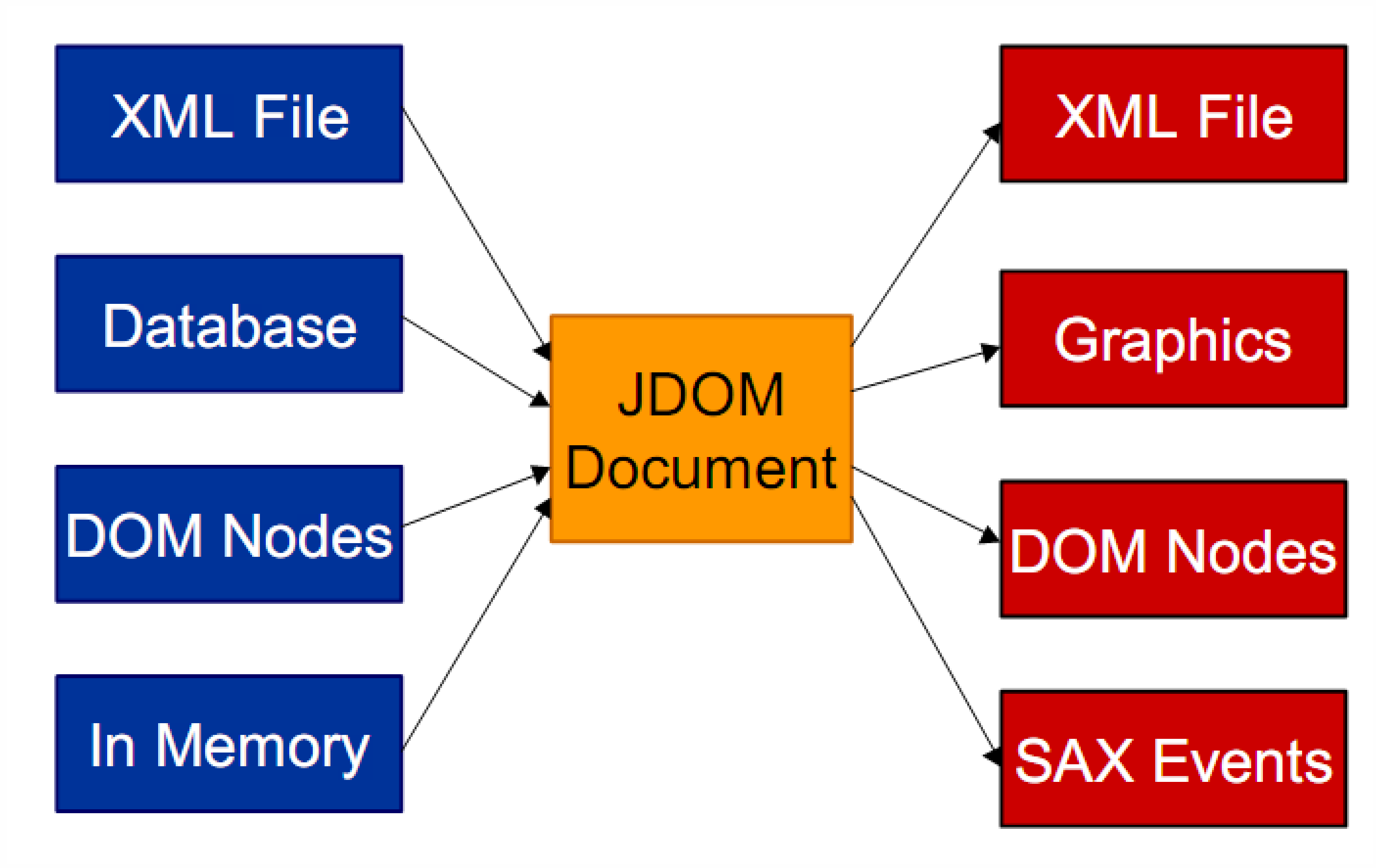
The JDOM Classes

- 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



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);`

Parsing a Document with JDOM

- Construct an `org.jdom.input.SAXBuilder` or an `org.jdom.input.DOMBuilder`; no parser specific code is needed
- Invoke the builder's `build()` method to build a `Document` object from a
  - `Reader`
  - `InputStream`
  - `URL`
  - `File`
  - `String` containing a SYSTEM ID
- If there's a problem building the document, a `JDOMException` is thrown
- Work with the resulting `Document` object

Navigation of 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");
```

Attributes

- **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();
```

CDATA 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();
  ```

CDATA content

- Text content can be changed directly
  Special chars are interpreted correctly

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

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