# Java™ API for XML Processing Samples

**Specification Version:** 1.2
**Reference Implementation (RI) Version:** 1.2.0-FCS

This release includes XML data and example programs showing how to use JAXP to process XML. Additional examples can be found on the http://xml.apache.org site.



The example programs include a cross-platform ant build file that can be used to build and run the example. Ant is a build tool similar to `make` on Unix and `nmake` on WindowsNT that is also an XML application. To use ant, download it from the website and read the install docs. Alternatively, you can also view the ant `build.xml` file to see what needs to be done to manually compile and run an example program on your platform.

> **Note:**
> The ant utility uses the value of the `JAVA_HOME` environment variable to determine which Java platform it uses to compile and run the sample scripts. Make sure that variable points to the version you intend to use. If using version 1.4, make sure that the JAXP jar files are installed, as described in the Release Notes.

---

## Sample XML Files

A handful of sample XML files have been provided in the "samples" subdirectory. Note that the links may not work depending on your browser environment. Please look in `../samples/data` if the links do not display in your browser.

- A simple Purchase Order (`book-order.xml`) suggesting how an on-line business might send and receive data.
- The original XML specification (`REC-xml-19980210.xml`) was written in XML. With a prepublication version of its Document Type Definition (DTD) file (`spec.dtd`), it's included here as a rather sophisticated example of how DTDs are used.
- Courtesy of Fuji Xerox, a short XML file in Japanese, using Japanese tags. This is a weekly report (`weekly-euc-jp.xml`) with its DTD (`weekly-euc-jp.dtd`), which can be validated.
- From a large collection of XML sample documents at the sunsite.unc.edu FTP server, The Two Gentlemen of Verona (`two_gent.xml`) and The Tragedy of Richard the Third (`rich_iii.xml`). These include their DTD (`play.dtd`).
- A simple example showing the use of XML namespaces (`namespace.xml`).
- A personal-schema.xml file containing some data that can be validated using its schema definition, personal.xsd.

---

## Printing a DOM Tree

One of the first things many programmers want to know is how to read an XML file and generate a DOM Document object from it. Use the DOMEcho example to learn how to do this in three steps. The important lines are:

```
// Step 1: create a DocumentBuilderFactory and setNamespaceAware
DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
dbf.setNamespaceAware(true);

// Step 2: create a DocumentBuilder
DocumentBuilder db = dbf.newDocumentBuilder();

// Step 3: parse the input file to get a Document object
Document doc = db.parse(new File(filename));
```

The program also gives an example of using an error handler and of setting optional configuration options, such as validation. Finally, this program helps you understand how DOM works by showing you the structure and contents of a DOM tree.

---

## SAX Program to Count Tags

The SAXLocalNameCount program counts the number of unique element local names in an XML document, ignoring the namespace name for simplicity. This example also shows one way to turn on DTD or XSD validation and how to use a SAX ErrorHandler.

There are several ways to parse a document using SAX and JAXP. We show one approach here. The first step is to bootstrap a parser. There are two ways: one is to use only the SAX API, the other is to use the JAXP utility classes in the javax.xml.parsers package. We use the second approach here because at the time of this writing it probably is the most portable solution for a JAXP compatible parser. After bootstrapping a parser/XMLReader, there are several ways to begin a parse. In this example, we use the SAX API.

## Schema Examples

Both of the sample programs include an option (`-xsd`) that lets you validate the incoming document using XML Schema, instead of the document's DTD. In addition, they include an `-xsdss` option that lets you specify the "schema source" (the file that defines the schema for the document).

Both programs define the following constants:

```
static final String JAXP_SCHEMA_LANGUAGE =
    "http://java.sun.com/xml/jaxp/properties/schemaLanguage";
static final String W3C_XML_SCHEMA =
    "http://www.w3.org/2001/XMLSchema";
static final String JAXP_SCHEMA_SOURCE =
    "http://java.sun.com/xml/jaxp/properties/schemaSource";
```

The schema language property defines the language the schema is written in. The W3C XML Schema

language is specified in these examples. The schema source property directs the parser to a schema to use, regardless of any schema pointer that the XML instance document may contain.

This code is abstracted from the SAX example:

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

// Set namespaceAware to true to get a parser that corresponds to
// the default SAX2 namespace feature setting.  This is necessary
// because the default value from JAXP 1.0 was defined to be false.
spf.setNamespaceAware(true);
spf.setValidating(true);

SAXParser saxParser = spf.newSAXParser();

// Set the schema language if necessary
try {
    saxParser.setProperty(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);
} catch (SAXNotRecognizedException x) {
    // This can happen if the parser does not support JAXP 1.2
    ...
}
...
saxParser.setProperty(JAXP_SCHEMA_SOURCE, new File(schemaSource));
```

And here is the code abstracted from the DOM example:

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

// Set namespaceAware to true to get a DOM Level 2 tree with nodes
// containing namesapce information.  This is necessary because the
// default value from JAXP 1.0 was defined to be false.
dbf.setNamespaceAware(true);

dbf.setValidating(true);
try {
    dbf.setAttribute(JAXP_SCHEMA_LANGUAGE, W3C_XML_SCHEMA);
} catch (IllegalArgumentException x) {
    // This can happen if the parser does not support JAXP 1.2
    ...
}
// Specify other factory configuration settings
dbf.setAttribute(JAXP_SCHEMA_SOURCE, new File(schemaSource));
...
DocumentBuilder db = dbf.newDocumentBuilder();
```

Note that the values are used to modify the SAX *parser*, using `setProperty()`, but they are used to modify the DOM parser *factory*, using `setAttribute()`.