

Install and configure Xalan-Java

Add XSL transformations to Java apps

Skill Level: Introductory

[Nicholas Chase](mailto:nicholas@nicholaschase.com) (nicholas@nicholaschase.com)

Author

26 Nov 2002

This tutorial is for developers who need to add XSL transformations to their Java applications. It describes the process of installing, configuring, and testing the Xalan-Java XSL Transformation engine, version 2.4.1.

Section 1. Introduction

Should I take this tutorial?

This tutorial is for developers who need to add XSL transformations to their Java applications. It describes the process of installing, configuring, and testing the Xalan-Java XSL Transformation engine, version 2.4.1.

This tutorial covers only the installation of Xalan and not its use. To understand and work through this tutorial, familiarity with Java or XSL is not required.

What is this tutorial about?

Extensible Stylesheet Language (XSL) allows for easy transformation of XML into other formats, including other XML documents. An XSL processor is necessary to perform these transformations.

This tutorial describes the process necessary for installing, configuring, and testing

the Xalan-Java XSL Transformation processor. Xalan, used in IBM products such as WebSphere Application Server, is the descendant of LotusXSL, created by Lotus and IBM and maintained by the original developers and the Apache Project.

Installing Xalan-J involves preparing the environment by obtaining an appropriate Java Virtual Machine, setting environment variables, and installing the files. After installation, you can set other environment variables, such as `CLASSPATH`.

Tools

The tools necessary for this tutorial depend greatly on the platform and existing or installed software.

- **A Java Virtual Machine:** A JVM, such as the IBM Java machine or Sun's JDK, must be installed and working on the target machine. Links to JVMs for various platforms are listed in [Resources](#).
- **The Xalan-Java 2.4.1 binary files:** Apache provides pre-compiled files, so you don't need the source files. Download the binaries from <http://apache.osuosl.org/xml/xalan-j/>.

Section 2. What is Xalan-Java?

Why transform XML files?

While it's common to build applications that directly process XML information, it is also common to simply transform one set of XML data into another set of XML data with a different structure, or into another form entirely. Common uses of transformation include transforming XML into HTML (or vice-versa), and taking XML from one system or application and transforming it before it arrives at a different system or application.

While applications can perform these transformations on their own, it is usually much easier to use an Extensible Stylesheet Language style sheet (XSL style sheet) to specify the transformation and then use a processor such as Xalan to perform the transformation.

XSLT style sheets

XSLT style sheets are built using templates. Each template specifies the content to transform and the rules for completing the transformation, taking a pattern-matching approach. For example, the following style sheet tells the processor to convert an XML document to an HTML document, converting all <heading> elements in the XML document to <h1> elements in the HTML output, and <paragraph> elements of the XML document to HTML <p> elements.

```
<?xml version="1.0"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="1.0">

  <xsl:template match="/">
    <html>
    <head>

    </head>
    <body>
      <xsl:apply-templates/>
    </body>
    </html>
  </xsl:template>

  <xsl:template match="heading">
    <h1><xsl:value-of select="." /></h1>
  </xsl:template>

  <xsl:template match="paragraph">
    <p><xsl:value-of select="." /></p>
  </xsl:template>
</xsl:stylesheet>
```

For more information on building XSLT style sheets, see [Resources](#).

Section 3. Prepare the environment

Install a Java Virtual Machine

Xalan-J is written entirely in Java code, and as such requires Java versions 1.1.8, 1.2.2, or 1.3. Users of Java 2 version 1.4.x need to make use of the [The Endorsed Standards Override Mechanism](#).

Download locations depend on the target platform:

- Download Windows, Linux, AIX, AS/400, OS/2, or OS390 versions from <http://www.ibm.com/developerworks/java/jdk/index.html>.
- Download Windows, Linux x86, or Solaris/SPARC x86 versions from <http://java.sun.com/j2se/1.3/>.
- Download a Mac OS X version from <http://devworld.apple.com/java/>.
- For other platforms, see <http://java.sun.com/cgi-bin/java-ports.cgi>.

Obtain the binaries

Different versions of the Xalan files are available depending on the developer's tolerance for instability. The current release at the time of this writing (2.4.1) can be downloaded at <http://apache.osuosl.org/xml/xalan-j/>.

Download the appropriate archive and place it in the location where you want the eventual Xalan directory to go.

Environment

To most efficiently use Xalan-J, it helps to have the Java executable on the system path, as determined by the `PATH` environment variable. (An environment variable is a variable that is available to all applications.) How this is accomplished is determined by your operating system.

Windows:

```
set path=%PATH%;c:\jdk1.3.1\bin
```

Make sure to use the actual location of the `java.exe` file. To make this setting apply to every opened window, add this command to `autoexec.bat` and restart your system. Alternatively, on Windows NT, choose `Start/Settings/Control Panel/System` and click the `Environment` tab, as shown in the figure at the right. In the upper box, click `Path` and add the new location in the bottom text field. Click `Set`, and then `OK`.

UNIX/Linux:

The proper way to set a variable depends on the type of UNIX and shell. To add the Java binaries to the path, type:

```
setenv path=$PATH:/path/to/add
```

or

```
set PATH=$PATH:/path/to/add
```

or

```
EXPORT PATH=$PATH:/path/to/add
```

For the value to be present after a reboot, make sure to add the path in the appropriate configuration file in `/etc`.

Section 4. Install the files

The installation process

Installing Xalan-J is a straightforward process involving the following steps:

1. [Choose a location](#)
2. [Unpack the files](#)
3. [Set the CLASSPATH](#)
4. [Choose a parser and processor](#)

Choose a location

The actual location of the Xalan-J installation is not necessarily important, as long as it is not inconvenient. If a structure of Java applications already exists, however, it may be convenient to install the files into this structure. All Xalan-J files are installed within a single folder; no information is written to the Windows registry. They are simply Java classes that may be used by other applications.

Unpack the files

The Xalan-J classes are distributed in ZIP format (`xalan-j_2_4_1-bin.zip`) or as a gzipped tarball (`xalan-j_2_4_1.tar.gz`). Before unpacking the file, place it in the *root* folder where the Xalan directory structure is to reside. For example, to install Xalan-J in `c:\xalan-j_2_4_1`, copy the `xalan-j_2_4_1.zip` file to `c:\` and unpack it

there.

In a Windows environment, a separate ZIP utility is not required to unpack the ZIP files, though it may be used. The `jar` utility, included with the Java Development Kit, can be used to unpack the files. The command to accomplish this is:

```
jar xf xalan-j_2_4_1-bin.zip
```

For a gzipped tarball, use the `tar` utility. For example:

```
tar xzvf xalan-j_2_4_1.tar.gz
```

or

```
gzip -d xalan-j_2_4_1.tar.gz | tar xvf -
```

The result is a directory called `xalan-j_2_4_1`. This directory may be renamed, but it is often better to leave it as-is to avoid confusion as to installed versions.

Included files

The Xalan-J distribution contains all of the files necessary to transform files. It also includes documentation and samples:

- **XML parser:** Xalan-J, as an XSLT processor, uses an XML parser. Although it is designed to be used with the XML parser of the developer's choice, by default it uses the Xerces parser. To avoid version problems, Xalan-J includes Xerces-Java 2.2.0 in the `xercesImpl.jar` file listed below.
- **bin:** Contains all of the *.jar files necessary to run various features of Xalan-J:
 - `BCEL.jar`
 - `bsf.jar`
 - `java_cup.jar`
 - `JLex.jar`
 - `regexp.jar`

- `runtime.jar`
- `xalan.jar`
- `xalansamples.jar`
- `xalanservlet.jar`
- `xercesImpl.jar`
- `xml-apis.jar`
- `xsltc.jar`

In general, only `xalan.jar`, `xercesImpl.jar`, `xml-apis.jar`, and `bsf.jar` are necessary.

- **docs:** Contains documentation on all of the relevant APIs, including both Xalan- and Xerces-specific classes and general XML classes, such as the DOM and SAX classes.
- **samples:** Contains source code and sample documents for the sample applications. This code may be examined and modified for the developer's own use.
- **License:** This text file contains the Apache license for Xalan-J, which permits both use and redistribution.
- **Keys:** This file contains encryption keys that are used for code signing.
- **readme.html:** This file simply redirects the user to the `docs/whatsnew.html` page, which leads into the documentation.

Set the CLASSPATH

To make a class available to a Java application, the application must know where to find it. Earlier versions of Java required the setting of a `CLASSPATH` environment variable, which told the `java` executable where to find these files.

Current versions no longer require the `CLASSPATH` variable, but it remains useful in situations where files are contained in other directories, such as the `xalan-j_2_4_1/bin` directory. Using the same method that's used to set the `PATH` variable for the Java executable (in [Environment](#)), set the `CLASSPATH` variable so that it includes the `xalan.jar`, `xercesImpl.jar`, and `xml-apis.jar` files.

For Java 1.3, however, it doesn't end there. The `java` executable will automatically use any classes located in `lib/ext` before it uses classes specified in the `CLASSPATH`, so be sure to check for outdated versions of files such as `xerces.jar`, `xercesImpl.jar`, `crimson.jar`, or `xml.jar`. Either remove

these files, or overwrite them with more current versions.

The Endorsed Standards Override Mechanism

An earlier, not entirely bug-free version of Xalan-J is also part of Java 1.4. Under normal circumstances, this is used instead of the current Xalan distribution, no matter what is on the system CLASSPATH.

Fortunately, these classes are *endorsed standards*, so you can override the Java version using the Endorsed Standards Override Mechanism by placing the relevant jar files (in this case, `xalan.jar`) in the appropriate directory. The default location is:

```
%JAVA_HOME%\lib\endorsed
```

on Windows, or:

```
%JAVA_HOME%/lib/endorsed
```

on a Linux system.

Developers can also choose a new location by setting the `java.endorsed.dirs` system property.

Users of older versions of Java need not be concerned with this issue.

Choose a parser and processor

As installed, Xalan-J uses the Xerces-J parser and the Xalan-J XSLT processor, but the design uses system properties to determine the appropriate classes, so developers can plug in different classes if desired.

The properties and their original values are:

```
javax.xml.transform.TransformerFactory
    = org.apache.xalan.processor.TransformerFactoryImpl

javax.xml.parsers.DocumentBuilderFactory
    = org.apache.xerces.jaxp.DocumentBuilderFactoryImpl

javax.xml.parsers.SAXParserFactory
    = org.apache.xerces.jaxp.SAXParserFactoryImpl
```

There are three ways to set these properties. In order of complexity, they are:

- **Set the system property in `jaxp.properties`:** The `java` application will check for properties files in `%JAVA_HOME%/lib`, where `%JAVA_HOME%` is the location of the Java installation. For example, to use the Crimson SAX parser rather than the Xerces SAX parser, create a file that reads:

```
javax.xml.parsers.SAXParserFactory=org.apache.crimson.jaxp.SAXParserFactoryImpl
```

- **Set the value from the command line:** Just as a `java` application looks for a `CLASSPATH` information command line, it also looks for system properties. They should be preceded by `-D`, as in the following line, which runs an application called `SimpleApp`:

```
java -Djavax.xml.parsers.SAXParserFactory  
org.apache.crimson.jaxp.SAXParserFactoryImpl SimpleApp
```

Note that there is no space between `-D` and the property name.

- **Build Xalan-J with the new value:** Working with the source files, change the entry for the appropriate property in `src/META-INF/services` and recompile.

Section 5. Test the installation

The sample applications

The best way to test the Xalan-Java installation is to attempt to run one of the sample applications included with the distribution. If these applications run successfully, Xalan is correctly installed and configured.

The samples demonstrate the various capabilities available within Xalan:

- **SimpleTransform:** This application transforms `birds.xml` into `birds.out` using `birds.xsl`.
- **UseStylesheetPI:** This application demonstrates embedded style sheets by using the style sheet processing instruction within an XML file to determine which XSL style sheet to use.

- **UseStylesheetParam:** This application takes a command-line parameter and uses it to populate a parameter within the style sheet.
- **SAX2SAX:** This application demonstrates the transformation of a SAX stream.
- **DOM2DOM:** This application demonstrates the transformation of an in-memory DOM document, providing an in-memory DOM document for further processing.
- **Pipe and UseXMLFilters:** These applications demonstrate methods for chaining transformations, where the results of one are the source for the next.
- **ApplyXPath:** This application provides an easy way to determine the content returned by a specific XPath expression.
- **AppletXMLtoHTML:** This application demonstrates the use of an applet to manipulate XML in the browser.
- **servlet:** This is actually a collection of four Java servlets and Java Server Pages that demonstrate the use of server-side XML processing to provide HTML results.
- **extensions:** This is a collection of examples demonstrating JavaScript and Java extensions to XSL, where code is executed as part of style sheet processing.
- **Trace:** This application demonstrates the use of TraceListener and TraceManager as an aid in debugging.
- **Validate:** These applications demonstrate ways to validate files in the context of transformations, as opposed to parsing.
- **trax:** This application (called `Examples`) demonstrates the use of the Transformation API for XML (TrAX).
- **translets:** These applications demonstrate the use of compiled style sheets, and require additional files on the `CLASSPATH`. For more information, see `docs/samples.html#translets`.

These samples are valuable not only because they allow testing of the Xalan-J installation, but also because the source code is included with the distribution and developers can use them as the basis for new applications. In particular, they can be useful for learning how to perform specific actions using Xalan-J.

Running the samples

Before running the samples, you must make sure that the Java executable knows

where to find both the Xalan and Xerces base classes (in the `xalan.jar`, `xercesImpl.jar`, and `xml-apis.jar` files) and the compiled sample applications (in the `xalansamples.jar` file). The former should already be part of the `CLASSPATH` variable, so you have two options for making the sample applications available.

Your first option is to simply add `xalansamples.jar` to the `CLASSPATH` variable. This is the simplest option, but it has the potential to lead to confusion later if any new applications share class names with any of the samples.

Your second option is to specify the `CLASSPATH` at the time of execution. If the `java` executable sees the `-cp` switch, it uses that classpath information. Consequently, to run, for example, the `SimpleTransform` sample, go to the `xalan-j_2_4_1\samples\SimpleTransform` directory and type:

```
java -cp %CLASSPATH%;c:\xalan-j_2_4_1\bin\xalansamples.jar
SimpleTransform
```

The preceding code was split into two lines for easier viewing. In reality, it is a single line of code.

Be sure to use the actual location of the `xalansamples.jar` file.

For more information on specifics regarding the sample applications check the `docs/samples.html` file included with the distribution.

Troubleshooting

The most common problem encountered in running the sample applications is the inability to locate a particular class. If problems arise, take the following steps:

1. Verify that the `java` executable is on the path, or that it is directly specified.
2. Verify that `xalan.jar`, `xercesImpl.jar`, and `xml-apis.jar` are part of the `CLASSPATH`. They are in the `bin` directory.
3. Check that no out-of-date files precede the above `*.jar` files in the `CLASSPATH`.
4. Make sure that no out-of-date files are in Java's `lib/ext` directory.
5. Make sure that `xalansamples.jar` is part of the `CLASSPATH`, or is

specified using the `-cp` switch.

6. If using the `-cp` switch, verify that it contains an accurate reference to the appropriate `*.jar` files.
7. Verify that the names are spelled correctly, including upper- or lowercase letters.
8. Some of the samples require additional `*.jar` files on the `CLASSPATH`. See the documentation for the specific sample application.

Using Xalan-J from the command line

In addition to providing classes that may be built into applications, Xalan-J provides a standalone Java application for transforming XML files using XSLT style sheets. This application uses parameters to determine the source and destination files, the style sheet to use, and other information regarding the transformation. For example, to use Xalan to transform `scores.xml` to `scores.html` using `scores.xsl`, type:

```
java org.apache.xalan.xslt.Process -IN scores.xml -XSL scores.xsl  
-OUT scores.out -HTML
```

The `-HTML` switch tells the processor to create transitional HTML 4.0, allowing for elements such as `` and `
`, which are not, as far as XML goes, well-formed.

Section 6. Summary

Tutorial summary

Xalan-Java is a Java-based XSL Transformation processor. Installing Xalan-J involves obtaining the distribution, unpacking it into the desired location, and setting the appropriate environment variables.

The Xalan-J distribution also includes sample files that are handy for both testing the installation and for serving as the basis for new applications.

Resources

Learn

- Xalan API documentation is included with the distribution. It can also be found at <http://xml.apache.org/xalan-j/apidocs/index.html>.
- Find out what XSLT is for and why it was designed as it is in Michael Kay's article, "[What kind of language is XSLT?](#)" (*developerWorks*, February 2001).
- For more information on getting started with Xalan-Java, see <http://xml.apache.org/xalan-j/getstarted.html>.
- Read [G. Ken Holman's article on XML.com](#) for an in-depth discussion of XSLT.
- Look into the official XSL spec and related resources on the World Wide Web Consortium's Web site at <http://www.w3.org/Style/XSL/>.
- Read [Benoit Marchal's tip](#) on using SAXTransformerFactory to achieve greater flexibility and convenience with XSLT (*developerWorks*, August 2001), or his feature on "[SAX, the Power API](#)" (*developerWorks*, August 2001).
- Get [developer information on WebSphere](#).
- Find out how you can become an [IBM Certified Developer in XML and related technologies](#).
- Explore many more XML resources on the [developerWorks XML zone](#).
- Stay current with [developerWorks technical events and Webcasts](#).

Get products and technologies

- Download Xalan-Java from the Apache project at <http://xml.apache.org/xalan-j/index.html>.
- Download Xalan C++ from the Apache project at <http://xml.apache.org/xalan-c/index.html>.
- Download versions of Java for various OSes from <http://www.ibm.com/developerworks/java/jdk/index.html>.
- Download Windows, Linux x86, and Solaris/SPARC x86 versions of Java from <http://java.sun.com/j2se/1.3/>.
- Download a Mac OS X version of Java from <http://devworld.apple.com/java/>.
- Build your next development project with [IBM trial software](#), available for download directly from developerWorks.

Discuss

- [Participate in the discussion forum for this content.](#)

About the author

Nicholas Chase

Nicholas Chase has been involved in Web site development for companies such as Lucent Technologies, Sun Microsystems, Oracle, and the Tampa Bay Buccaneers. Nick has been a high school physics teacher, a low-level radioactive waste facility manager, an online science fiction magazine editor, a multimedia engineer, and an Oracle instructor. More recently, he was the Chief Technology Officer of Site Dynamics Interactive Communications in Clearwater, Florida, USA, and is the author of three books on Web development, including *XML Primer Plus* (Sams). He loves to hear from readers and can be reached at nicholas@nicholaschase.com.