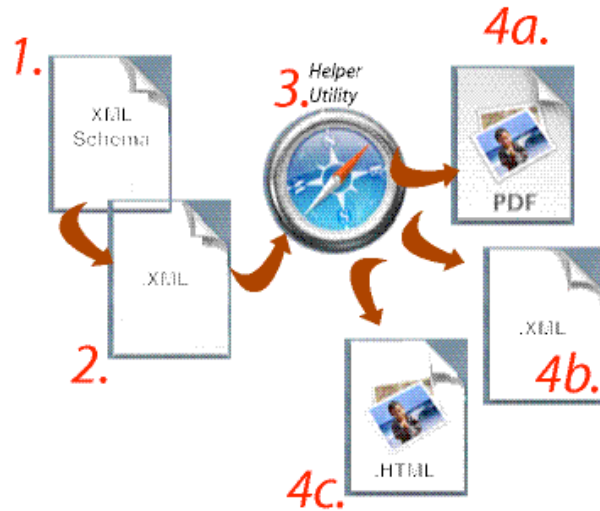


## CREATING XML RECORDS AND PROCESSING THEM FOR THE NET



**XML** see also <http://www.w3.org/Style/XSL> or <http://www.zvon.org/xxl/XSLTreference/Output>

XML is used to provide structure to flat-file documents. The structure is outlined in a DTD or an XML Schema. These rules then guide the creation of an .xml file. XML files can be subjected to stylistic changes (like a css), or generate meaningful subsets of re-arranged data (like a rdms).

We need: (1) a source .xml document, (2) *application* of the XSL stylesheet to the *source* document, (3) a computer program to compile the xml document with the xsl stylesheet (or a "transformation"), to get a (4) *result* document. XSL is 3 parts: XSLT (XSL transformations), XPath (to look for a specific part in the xml document), and XSL-FO (looks similar to a cascading style sheet (.css); e.g., <fo:block font-weight="bold">

### Examples

Source document:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- OldData.xml -->
<OldData>
  <Name>Chan</Name>
  <Zip>40506</Zip>
</OldData>
```

XSL Stylesheet:

```
<?xml version="1.0"?>
<!-- old2new.xsl -->
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="xml" indent="yes"/>
  <xsl:template match="/">
    <xsl:comment>NewData.xml</xsl:comment>
    <xsl:element name="NewData">
      <xsl:element name="LastName">
        <xsl:value-of select="//Name"/>
      </xsl:element>
      <xsl:element name="PostalCode">
        <xsl:value-of select="//Zip"/>
      </xsl:element>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

Result document:

```
<?xml version="1.0" encoding='UTF-8'?>
<!--the new data, in NewData.xml -->
<NewData>
  <LastName>Chan</LastName>
  <PostalCode>40506</PostalCode>
</NewData>
```

Program to Apply XSL:

```
import javax.xml.transform.TransformerFactory;           // LOAD THE LIBRARIES
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.stream.StreamSource;
import javax.xml.transform.stream.StreamResult;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;

public class transformFile {
    public static void main(String[] args)
        throws TransformerException, FileNotFoundException {
        TransformerFactory factory;           // Build a "factory"
        Transformer transformer;             // to process "something"

        File oldFile, xslFile, newFile;
        StreamSource oldStream, xslStream;
        StreamResult newStream;

        factory = TransformerFactory.newInstance();
        if (args.length == 3) {
            oldFile = new File(args[0]);           // get the .xml file (source)
            oldStream = new StreamSource(oldFile);

            xslFile = new File(args[1]);           // get the .xsl file
            xslStream = new StreamSource(xslFile);

            newFile = new File(args[2]);
            newStream = new StreamResult(newFile);           // output an .xml file (result)

            transformer = factory.newTransformer(xslStream);
            transformer.transform(oldStream, newStream);
        }
        else {
            System.out.println("Usage: java transformIt ");
            System.out.println("old.xml xform.xsl new.xml");
        }
    }
}
```

**Viewing XML Data on the Web** .xml file (See below note #1)

```
<?xml version = "1.0" encoding="UTF-8"?>
<!--book.xml -->

<Chapter number = "1" title = "Welcome">
  <Section title="Starting class">
    <Subsection title="Lesson 1"/>
  </Section>

  <Section title="Turning on the computer">
    <Subsection title="red switch"/>
    <Subsection title="Boot"/>
    <Subsection>cold boot</Subsection>
    <Subsection>warmboot</Subsection>
  </Section>
</Chapter>
```

**Creating an HTML File from XML Data:**

```
<?xml version="1.0"?>
<!-- file name is Formal.xml -->

<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output method="html">
  <xsl:template match="Chapter">
    <h2>
```

```

Chapter
  <xsl:value-of select="@number"/>
  <xsl:value-of select="@title"/>
</h2>
<xsl:apply-templates />
<xsl:template>

<xsl:template match="Section">
  <h3>
    <code>&#160;&#160;&#160;</code>
    <xsl:value-of select="@title"/>
  </h3>
</xsl:template>

<xsl:template match="Subsection">
  <br>
  <code>&#160;&#160;&#160;&#160;&#160;&#160;</code>
  <xsl:value-of select="."/>
  <br/>
</xsl:template>
</xsl:stylesheet>

```

**Note #1** – Internet Explorer will merge the xml sheet with the stylesheet if you create an .xml document (like the one above), but replace the first line with this one:

```
<?xml-stylesheet type="text/xsl" href="Formal.xsl"?>
```

### Java Servlet to Create HTML pages from XML

```

import javax.xml.transform.TransformerFactory;           // Load the xml transformer libraries
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.stream.StreamSource;
import javax.xml.transform.stream.StreamResult;
import java.io.File;
import java.io.PrintWriter;
import java.io.IOException;

import javax.servlet.http.HttpServlet;                   // load libraries for servlet
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class transformFile extends HttpServlet {

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws IOException {
        response.setContentType("text/html");           // generate html, not xml
        String docName=request.getParameter("docName"); // get name of .xml file
        String sheetName = request.getParameter("sheetName"); // get stylesheet to use

        TransformerFactory factory;
        Transformer transformer;

        File oldFile, xslFile;
        PrintWriter out;

        StreamSource oldStream, xslStream;
        StreamResult newStream;

        factory = TransformerFactory.newInstance();

        try {
            oldFile = new File(docName + ".xml");
            oldStream = new StreamSource(oldFile);
            xslFile = new File(sheetName + ".xsl");
            xslStream = new StreamSource(xslFile);

            out = response.getWriter();                  // get ready to send back to user
            newStream = new StreamResult(out);

```

```

        transformer = factory.newTransformer(xslStream);           // do the transformation
        transformer.transform(oldStream, newStream);             // add your own code for output
    }
    catch (TransformerException e {
        System.out.println(e.getMessage());
    }
}
}
}

```

The url for this would look like:

<http://localhost:8080/examples/servlet/transformIt?docName=chapter?sheetName=Formal>

Other points:

Parsers use the following (and other) APIs:

DOM, SAX, JDOM, Xerces, ElectricXML. Note that if you use JDK 1.4 and Jakarta-Tomcat 4 there is a ClassCasdt conflict. Both xerces.jar and dom (or maybe sax) have the same class names, so remove the xerces.jar from the web server's folder (and any path or CLASSPATH statements) and you'll be okay.

**Binding with JAXB:** combines the DTD with XML to create (or "generate") Java class files (e.g., if the DTD has a "Total" element, the schema compiler will create a "getTotal()" and "setTotal()" classes!

*Final example:*

### A SAX Parser:

*the source .xml document*

```

<?xml version="1.0" encoding="UTF-8"?>
<!--file name: mySource.xml -->
<Clients>
  <Client id = "100">BixCo</Client>
  <Client>Fred Jones Inc</Client>
  <Client>Fifi Enterprises</Client>
</Clients>

```

*The parser:*

```

import javax.xml.parsers.SAXParserFactory;           // load libraries for SAX
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.ParserConfigurationException;

import org.xml.sax.XMLReader;                       // load libraries for XML
import org.xml.sax.SAXException;

import java.io.File
import java.io.IOException;

public class CallSAX {
    static public void main(String[] args) throws SAXException,
        ParserConfigurationException, IOException {
        SAXParserFactory factory = SAXParserFactory.newInstance();           // build the factory...
        SAXParser saxParser = factory.newSAXParser();                         // bring in the tools
        XMLReader xmlReader = saxParser.getXMLReader();                       // build XML w/ the tools

        // note this MyHandler is not shown
        xmlReader.setContentHandler(new MyHandler());
        try {
            xmlReader.parse(new File(args[0]).toURL().toString());
        } catch (SAXException s) {
            System.out.println("Error! Yikes!");
            s.printStackTrace();
        }
    }
}

```