Java XML by Example (Part I): JAXP

Author: **Darshan Singh** (Managing Editor, PerfectXML.com)
Last Updated: July 11, 2003

Java promises "portable code" and XML assures "portable data". These two technologies can be used together to architect enterprise application that work cross-platform and over the Internet. Sun Microsystems and various other organizations are continuously working to add and enhance XML/Web services support in Java.

The goal of this articles series is less talk and more code, with focus on a particular Java XML technology. This time, we'll focus on **Java API for XML Processing (or JAXP)**.

- **Loading an XML Document using DOM**: First example illustrates loading an XML document into a DOM tree and checking if the XML document is **well-formed**.

- **Creating an XML Document using DOM**: Second example illustrates using DOM to create an XML document from scratch and saving it to a disk file. This example converts a comma separated (CSV) file to an XML document.

- **Evaluate XPath Expressions**: XML Path Language (or **XPath**) is a W3C standard that primarily allows identifying parts of an XML document. **Click here** for a quick introduction to XPath.

  The third example uses **SAXON 6.5.2** alongwith JAXP to load an XML document using DOM and then evaluate an XPath expression.

- **Applying XSLT Stylesheets**: XSL Transformations (or **XSLT**) stylesheets are used to *transform* XML documents into any other text format (such as XML, HTML, WML, etc.).

  The fourth example accepts names of XML and XSLT files as command line parameters, applies the transformation and sends the transformed output to the console.

- **Calling a .NET Web Service**: XML Web services created using ASP.NET support GET and POST methods of Web service invocation in addition to SOAP. That means, a Web service method can be called by sending a HTTP GET request or a HTTP POST request, in addition to SOAP invocation. This example uses SAX to load a remote XML document - the URL provided in essence is a HTTP GET request to a Web service. This example applies XSLT stylesheet on the XML returned by the Web service and saves the transformed HTML into a disk file.

We have used **J2SE 1.4.1_02** and **SAXON** in the following examples; and **Eclipse** as the IDE to write and debug these Java console applications.

Example 1: Is Well-Formed?

The following example is a console application that accepts XML file name/URL and tries to load this XML document. If the parser succeeds in loading/parsing the XML document, without raising any exception, we can assume that the XML document is well-formed. It uses `DocumentBuilderFactory` to create an instance of `DocumentBuilder` class on which then we call the parse method passing it the specified XML URI.

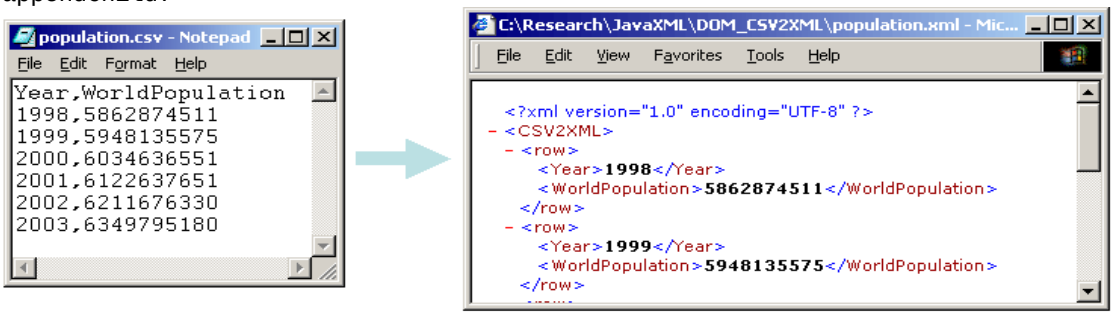
Loading an XML document using DOM (IsWellFormed.java)

```
/*
    DOM Example: Check if provided URI is a well-formed XML document
    Illustrates: Loading XML document using DOM
*/
import javax.xml.parsers.*;

public class IsWellFormed {

    public static void main(String[] args)
    {
        if(args.length != 1)
        {
            System.err.println("Usage: java IsWellFormed
<xmlFileNameOrUrl>\nExamples:");
            System.err.println("\n\tjava IsWellFormed c:\\1.xml");
            System.err.println("\n\tjava IsWellFormed
http://www.PerfectXML.com/books.xml");
            return;
        }

        try
        {
            DocumentBuilderFactory domFactory =
DocumentBuilderFactory.newInstance();
            DocumentBuilder domBuilder =
domFactory.newDocumentBuilder();
            domBuilder.parse(args[0]);

            System.out.println("'" + args[0] + "' is well-formed.");
        }
        catch(org.xml.sax.SAXException exp)
        {
            System.out.println("'" + args[0] + "' is NOT well-
formed.\n" + exp.toString());
        }
        catch(FactoryConfigurationError exp)
        {
            System.err.println(exp.toString());
        }
        catch(ParserConfigurationException exp)
        {
            System.err.println(exp.toString());
        }
        catch(Exception exp)
```

```
        {
            System.err.println(exp.toString());
        }
    }
}
```

Example 2: CSV to XML

This example reads a CSV file and creates an XML document from it. The first line in the CSV file is assumed to be the field/column names, and these names are used as element names in the XML document. Like in previous example, `DocumentBuilderFactory` is used to create an instance of `DocumentBuilder` class. This time we call **`newDocument`** method on `DocumentBuilder`. This method returns an instance of `org.w3c.dom.Document` class that is then used to create XML document by calling methods such as `createElement` and `appendChild`.



Creating XML Document using DOM (CSV2XML.java)

```
/*
    DOM Example: Creating XML document
        Converts CSV file to XML
        First line in CSV file is field/column names -
           which is also used as element names while creating XML
document

*/
import java.io.*;
import java.util.*;
import org.w3c.dom.*;
import javax.xml.parsers.*;
import javax.xml.transform.*;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

public class CSV2XML {

    //  Protected Properties
    protected DocumentBuilderFactory domFactory = null;
    protected DocumentBuilder domBuilder = null;

    //  CTOR
    public CSV2XML()
```

```
    {
        try
        {
            domFactory = DocumentBuilderFactory.newInstance();
            domBuilder = domFactory.newDocumentBuilder();
        }
        catch(FactoryConfigurationError exp)
        {
            System.err.println(exp.toString());
        }
        catch(ParserConfigurationException exp)
        {
            System.err.println(exp.toString());
        }
        catch(Exception exp)
        {
            System.err.println(exp.toString());
        }
    }

    public int convertFile(String csvFileName, String xmlFileName)
    {
        int rowsCount = -1;
        try
        {
            Document newDoc = domBuilder.newDocument();

            //  Root element
            Element rootElement = newDoc.createElement("CSV2XML");
            newDoc.appendChild(rootElement);

            //  Read comma seperated file
            BufferedReader csvReader;
            csvReader = new BufferedReader(new
FileReader(csvFileName));
            int fieldCount = 0;
            String[] csvFields = null;
            StringTokenizer stringTokenizer = null;

            //  Assumption: first line in CSV file is column/field
names
            //  As the column names are used to name the elements in
the XML file,
            //  avoid using spaces/any other characters not suitable
for XML element naming
            String curLine = csvReader.readLine();
            if(curLine != null)
            {
                stringTokenizer = new StringTokenizer(curLine, ",");
                fieldCount = stringTokenizer.countTokens();
                if(fieldCount > 0)
                {
                    csvFields = new String[fieldCount];
                    int i=0;
```

```
                    while(stringTokenizer.hasMoreElements())
                        csvFields[i++] =
String.valueOf(stringTokenizer.nextElement());
                }
            }

            //  Now we know the columns, Let's now read data row lines
            while((curLine = csvReader.readLine()) != null)
            {
                stringTokenizer = new StringTokenizer(curLine, ",");
                fieldCount = stringTokenizer.countTokens();

                if(fieldCount > 0)
                {
                    Element rowElement = newDoc.createElement("row");

                    int i=0;
                    while(stringTokenizer.hasMoreElements())
                    {
                        try
                        {
                            String curValue =
String.valueOf(stringTokenizer.nextElement());
                            Element curElement =
newDoc.createElement(csvFields[i++]);

curElement.appendChild(newDoc.createTextNode(curValue));
                            rowElement.appendChild(curElement);
                        }
                        catch(Exception exp)
                        {
                        }
                    }

                    rootElement.appendChild(rowElement);
                    rowsCount++;
                }

            }

            csvReader.close();

            //  Save the document to the disk file
            TransformerFactory tranFactory =
TransformerFactory.newInstance();
            Transformer aTransformer = tranFactory.newTransformer();

            Source src = new DOMSource(newDoc);
            Result dest = new StreamResult(new File(xmlFileName));
            aTransformer.transform(src, dest);

            rowsCount++;
        }
        catch(IOException exp)
```

```
        {
            System.err.println(exp.toString());
        }
        catch(Exception exp)
        {
            System.err.println(exp.toString());
        }

        return rowsCount;
    }

    public static void main(String[] args)
        {
        try
        {
            if(args.length != 2)
            {
                System.out.println("Usage: java CSV2XML <inputCSVFile>
<outputXMLFile>");
                return;
            }
        }
        catch(Exception exp)
        {
            System.err.println(exp.toString());
        }

        try
        {
            CSV2XML csvConverter = new CSV2XML();
            int rowsCount = csvConverter.convertFile(args[0], args[1]);

            if(rowsCount >= 0)
            {
                System.out.println("CSV File '" + args[0] +
                        "' successfully converted to XML File '"+
args[1] + "'\n" +
                        "(" + String.valueOf(rowsCount) + " rows)");
            }
            else
            {
                System.out.println("Error while converting input CSV
File '" + args[0] +
                        "' to output XML File '"+ args[1] + "'");
            }
        }
        catch(Exception exp)
        {
            System.err.println(exp.toString());
        }
        }
}
```

The main function creates an instance of CSV2XML class and calls convertFile method. This

method starts reading the CSV file. It uses `stringTokenizer` to first get the column names (from the first row) and then the actual data items (for each row starting second row in the file) and creates XML elements.

The above code uses JAXP classes `TransformerFactory` and `Transformer` to save the document. The alternative approach would be to use `org.apache.crimson.tree.XmlDocument` class as below:

```
BufferedWriter bufferWriter =
    new BufferedWriter(new FileWriter(xmlFileName));

XmlDocument xDoc = (XmlDocument)newDoc;
xDoc.write(bufferWriter);

bufferWriter.close();
```

Example 3: Evaluate XPath Expressions

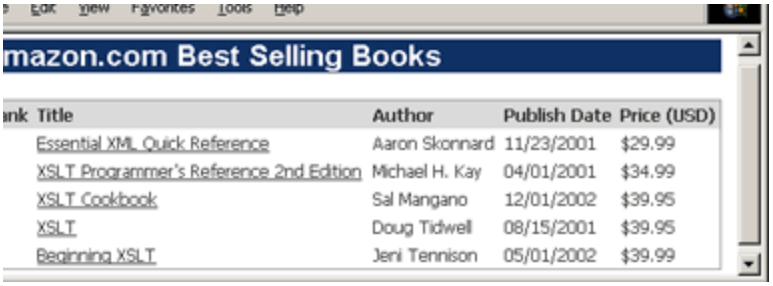
The following code uses JAXP DOM along with SAXON to load a local XML document, and evaluate an XPath expression. It first tries to tries to get result as a node-set. If XPath results in a node-set, **`NodeEnumeration`** is used to iterate over the nodes. SAXON throws **`XPathException`** if the expression does not result in a node-set (for example, XPath expression like `count(/options/option)`. In such cases, the following code uses **`evaluateAsString`** - which can also convert from integer or boolean to string.

Using SAXON alongwith JAXP to evaluate XPath Expressions (DOMSaxon_EvalXPath.java)

```
/*
   DOM XPath Example: Evaluate an XPath expression
   Uses SAXON alongwith JAXP DOM to evaluate an XPath expression
   Make sure SAXON jar is in the classpath
   Command line parameters are source XML URL and an XPath expression
   Example does not support XML Namespaces
*/
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.*;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;

import com.icl.saxon.expr.*;
import com.icl.saxon.Context;
import com.icl.saxon.om.*;

public class DOMSaxon_EvalXPath {
   public static void main(String[] args) {
      try
      {
         if(args.length != 2)
         {
            System.err.println("Usage java DOMSaxon_EvalXPath
<xmlFileNameOrUrl> " +
               "<XPathExpression>\nExamples:");
```

```
            System.err.println("java DOMSaxon_EvalXPath Options.xml " +
               "/options/option[@key='newFeedsUrl']/@value");

            System.err.println("java DOMSaxon_EvalXPath Options.xml
count(/options/option)");

            System.err.println("java DOMSaxon_EvalXPath Options.xml " +
"" +
               "\"concat(/options/option[position()=last()]/@key, '=',
" +
               "/options/option[position()=last()]/@value)\"");

            System.err.println("java DOMSaxon_EvalXPath
http://www.PerfectXML.com/books.xml " +
               "count(/catalog/book)");

            return;
         }
      }
      catch(Exception exp)
      {
         System.err.println(exp.toString());
      }

      //   Evaluate XPath expression now
      DOMSaxon_EvalXPath xpathEval = new DOMSaxon_EvalXPath();

      System.out.println(xpathEval.evalXPath(args[0], args[1]));
   }

   public String evalXPath(String xmlFileToLoad, String xpathExpr)
   {
      String result="";

      try
      {
         //   Setting a system property required by SAXON
         System.setProperty("javax.xml.parsers.DocumentBuilderFactory",
            "com.icl.saxon.om.DocumentBuilderFactoryImpl");

         //   Create DocumentBuilderFactory
         DocumentBuilderFactory factory =
DocumentBuilderFactory.newInstance();

         //   Load the specified XML document
         DocumentBuilder builder = factory.newDocumentBuilder();
         InputSource inputSrc = new InputSource
            (new FileInputStream(new File(xmlFileToLoad)));
         Node xmlDoc = builder.parse(inputSrc);

         //   Assign root node to SAXON NodeInfo, which is then used to
         //   set the current context node for XPath searches
         NodeInfo nodeInfo = (NodeInfo) xmlDoc;
```

```
         Context context = new Context();
         context.setContextNode(nodeInfo);

         //   If XML document defines namespaces, this would be the
time to declare them
         //   Use StandaloneContext.declareNamespace for this purpose
         DocumentInfo docInfo = nodeInfo.getDocumentRoot();
         NamePool namePool = docInfo.getNamePool();
         StandaloneContext standaloneContext = new
StandaloneContext(namePool);

         //   Parse the input XPath expression
         Expression expr = Expression.make(xpathExpr,
standaloneContext);

         try
         {
            //   First, try to get a node-set, if it fails then try to
            //   evaluate XPath expression as a string result value
            NodeSetValue resultSet = expr.evaluateAsNodeSet(context);

            //   Loop over result node-set
            NodeEnumeration enumResult = resultSet.enumerate();
            int countNodes = 0;
            while(enumResult.hasMoreElements())
            {
               try
               {
                  NodeInfo curResultNode = enumResult.nextElement();
                  String curNodeValue = curResultNode.getStringValue();

                  if(curNodeValue != null && curNodeValue.length() > 0)
                     result += curNodeValue + "\n";

                  countNodes++;
               }
               catch(Exception exp)
               {
                  exp.printStackTrace();
               }
            }

            if(countNodes > 0)
            {
               if(countNodes==1)
               {
                  result += "\nOne matching node found!";
               }
               else
               {
                  result += "\n" + String.valueOf(countNodes) +
                     " matching nodes found!";
               }
            }
```




```
            }
            catch(XPathException exp)
            {
               result =
expr.evaluateAsString(context);
            }
         }
         catch(SAXException exp)
         {
            System.out.println("'" + xmlFileToLoad + "' is NOT well-
formed.\n" + exp.toString());
         }
         catch(FactoryConfigurationError exp)
         {
            System.err.println(exp.toString());
         }
         catch(ParserConfigurationException exp)
         {
            System.err.println(exp.toString());
         }
         catch(Exception exp)
         {
            System.err.println(exp.toString());
            exp.printStackTrace();
         }

         return result;
      }
}
```

Example 4: Apply XSLT

This console application accepts names of local XML and XSLT files, uses JAXP classes `TransformerFactory` and `Transformer` to apply the transformation and send the output to the console. The code download for this article includes couple of XML and XSLT stylesheet files that you can try with this example.

Transform XML Document (ApplyXSLT.java)

```
/*
 * Apply XSLT stylesheets
 * Input parameters are names of local XML and XSLT files
 * Output is sent to the console
 * XSLT Parameters are not supported.
 */

import javax.xml.transform.*;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

import java.io.*;
```

```
public class ApplyXSLT {

   public static void main(String[] args) {
      int argsCount=0;
      try
      {
         argsCount = args.length;
         if(argsCount != 2)
         {
            System.err.println("Usage java ApplyXSLT <localXMLFile> " +
                    "<localXSLTFile>\nExamples");

            System.err.println("\tjava ApplyXSLT BestBooks.xml
BestBooks.xsl ");
            System.err.println("\tjava ApplyXSLT lowercase.xml
ToUpper.xsl ");
            System.exit(1);
         }
      }
      catch(Exception exp)
      {
         exp.printStackTrace();
         System.exit(1);
      }

      try
      {

         //   Source XML File
         StreamSource xmlFile = new StreamSource(new File(args[0]));

         //   Source XSLT Stylesheet
         StreamSource xsltFile = new StreamSource(new File(args[1]));
         TransformerFactory xsltFactory =
TransformerFactory.newInstance();
         Transformer transformer =
xsltFactory.newTransformer(xsltFile);

         //   Send transformed output to the console
         StreamResult resultStream = new StreamResult(System.out);

         //   Apply the transformation
         transformer.transform(xmlFile, resultStream);
      }
      catch(Exception exp)
      {
         exp.printStackTrace();
      }
   }
}
```

Example 5: .NET Web Service Client

This console application accepts two parameters – zipcode and name of the instrument you want to learn! The second parameter (instrument) is optional. Given these values, it then sends a **GET** request to an ASP.NET Web service named **SearchMusicTeachers**. The Web method returned XML is then transformed using XSLT and the resultant HTML is saved as a disk file.

Searching for a music teacher (SAX_CallDotNetWebSvc.java)

```
/*
 *   Console application that loads a remote XML document
 *   The URL provided is actually GET call to a .NET Web service
 *   The command line parameters:
 *      Zip code (ex: 60195, 98007, etc.)
 *      Instrument (Optional) (ex: Piano, Guitar, Drums, etc.)
 *
 *   Applies XSLT stylesheet on the loaded document and  saves the
 *   HTML output to a file named output.html.
 */
import java.io.*;
import java.net.URL;
import javax.xml.transform.*;
import javax.xml.transform.sax.SAXSource;
import javax.xml.transform.stream.*;
import org.xml.sax.*;

public class SAX_CallDotNetWebSvc {

   public static void main(String[] args) {
      try
      {
         if(args.length < 1)
         {
            System.err.println("Usage: java SAX_CallDotNetWebSvc
<zipCode>" +
                " [Instrument]\nExamples:");

            System.err.println("\tjava SAX_CallDotNetWebSvc 60195");
            System.err.println("\tjava SAX_CallDotNetWebSvc 60195
Piano");
            System.exit(1);
         }
      }
      catch(Exception exp)
      {
         exp.printStackTrace();
      }
      try
      {
         String instrument = "";
         if(args.length > 1 && args[1] != null)
            instrument = args[1];

         URL webSvcGetURL = new
URL("http://www.PerfectXML.net/Webservices" +
```

```
"/MusicTeachers/MusicTeachers.asmx/FindMusicTeachers?ZipCode=" +
            args[0] + "&Instrument=" + instrument +
"&SkillLevel=&Style=&Radius=" +
            "20&RestrictResultsCount=20");

         System.out.println("Loading " + webSvcGetURL + "\nPlease
wait...\n");
         BufferedReader bufferedReader = new BufferedReader
            (new InputStreamReader(webSvcGetURL.openStream()));

         //   XML
         SAXSource saxSource = new SAXSource(new
InputSource(bufferedReader));

         String curDir = new File(".").getCanonicalPath();
         String xslFileName = curDir + File.separator +
"MusicTeachers.xsl";
         String outputFileName = curDir + File.separator +
"output.html";

         //   XSLT
         StreamSource xlstStreamSource = new StreamSource(new
File(xslFileName));

         //   Output file
         File resultHTMLFile = new File(outputFileName);

         if(resultHTMLFile.exists())
            resultHTMLFile.delete();

         StreamResult streamResult = new StreamResult(resultHTMLFile);

         //   Get XML, apply Transformation, save results
         TransformerFactory factory = TransformerFactory.newInstance();
         Transformer transformer =
factory.newTransformer(xlstStreamSource);
         transformer.transform((Source)saxSource, streamResult);

         System.out.println("Results saved into " + outputFileName +
".");
      }
      catch(Exception exp)
      {
         exp.printStackTrace();
      }
   }
}
```

Summary

XML is now being used for varied applications, such for data transfer over Internet, configuration files, messaging over Internet, graphics (SVG), wireless and multimodal applications (WAP, WML, SALT), and so on. The textual nature of XML makes it highly portable. Many developers are including XML in their Java applications architecture design. The best way to learn Java XML features is by trying out some examples. In this article series, I'll show you five or more examples with focus on a particular Java XML technology. In this first part, you learned about JAXP (and SAXON) and how to use it for creating XML documents, selecting nodes, and applying XSLT stylesheets. Next article in this series will focus on SAX and how to program it in Java.