



Using STAX with peoplecode

[View](#) [Attachments \(1\)](#) [Info](#)

Added by [seshagiri_veerapaneni](#), last edited by [seshagiri_veerapaneni](#) on Sep 24, 2008 ([view change](#))
Labels: (None) [EDIT](#)

Warning

The following example is just a Proof of Concept (POC) intended to prove that customers can extend PeopleTools to make use of STAX. This code has not been tested in our labs and our support teams will not be able to answer any questions related to this code. No research has been done regarding the License terms of the dependent jar file and how it impacts the customer usage.

Note:- The views expressed on this document are my own and do not necessarily reflect the views of Oracle.

```
/*
Problem Statement:
PeopleTools allows xml processing using DOM which is not scalable for handling large XML documents.
Generally DOM documents need 10 to 20 times memory compared to size of the XML file .Ex: a 10mb xml file will need 100mb to 200mb of memory.
Also, once this memory is reserved PeopleTools process does not free this in anticipation to reuse the memory for further DOM objects .

Solution:
Generally documents which grow have repeated entries and the number if these entries increase the document grows .
The idea is using streams we should be able to load partial documents (each entry at a time) and process each entry independently.

There are currently two popular stemming API
1.SAX :- This is event based API where caller gets call backs when ever a new element , attribute .... is detected
This will be tough to implement as the program is with java and has to callback into PeopleCode
Also this will result in too many JNI callbacks which can cause performance problems.
2.STAX: JSR 173 , this is a pull based API where the caller controls when to process the next event
This is easier to integrate with PeopleCode most of the logic can be implemented in java with no callbacks.

Code:
This implementation needs two jar files
1.The STAX API (i downloaded sax-1_0-fr2-spec-apidocs.zip which has jsr173_1.0_api.jar from http://jcp.org/aboutJava/communityprocess/final/jsr1
2.STAX parser (i downloaded sjsxp_101.class which has sjsxp.jar from https://sjsxp.dev.java.net/ )

To compile / run the code use following
1.compile using "javac -classpath .;sjsxp.jar;jsr173_1.0_api.jar com\peoplesoft\staxpoc\*.java"
2.Can run/test the code using "java -classpath .;sjsxp.jar;jsr173_1.0_api.jar com.peoplesoft.staxpoc.STAXIBRowsetMessage %1"
3.Add the above jar files and classes to PS_HOME\classes directory to invoke these java classes from PeopleCode

*/
```

Information

Message Segments should be used when the sending system can implement it as this addresses both transmission and processing issues. This solution can be used when large xml file is available in file system.

This sample will need changes to make it work for your particular scenario .
Did not get enough time to add comments and planning to add this later.

Test Results:

I was able to process a 540mb file with around 220000 transactions with out updating the database on my dell laptop with Intel Core 2 Due T7300 2Ghz the process completed in 15min while using only 60mb of memory.

STAXRS.MAIN.Step01

```
Local JavaObject &joSTAXIBRowsetMessage;  
Local string &inrsxmlfile = "C:\seshabackup\casetemp\IB\sampleDataAndLogs\stax\USER_PROFILE.xml";  
Local string &header, &footer, &transaction;  
Local Message &MSG;  
Local Rowset &RS;  
  
&joSTAXIBRowsetMessage = CreateJavaObject("com.peoplesoft.staxpoc.STAXIBRowsetMessage", &inrsxmlfile);  
&header = &joSTAXIBRowsetMessage.getHeader();  
&footer = &joSTAXIBRowsetMessage.getFooter();  
  
If &header = "NULL" Then  
    throw CreateException(0, 0, "Error reading file or header");  
End-If;  
  
If &footer = "NULL" Then  
    throw CreateException(0, 0, "Error reading footer");  
End-If;  
  
&transaction = &joSTAXIBRowsetMessage.getNextTrasaction();  
  
While &transaction <> "NULL"  
    &transaction = &header | &transaction | &footer;  
  
    &MSG = CreateMessage(Operation.USER_PROFILE);  
    &MSG.LoadXMLString(&transaction);  
    &RS = &MSG.GetRowset();  
    /*Process you rowset here */  
    rem MessageBox(0, "", 0, 0, "Tetst:" | &RS.GetRow(1).GetRecord(Record."PSOPRDEFN").GetField(Field."OPRID").Value);  
  
    &transaction = &joSTAXIBRowsetMessage.getNextTrasaction();  
End-While;
```

STAXIBRowsetMessage.java

```
package com.peoplesoft.staxpoc;

import javax.xml.stream.XMLStreamReader;

public class STAXIBRowsetMessage {

    String header = null;
    String footer = null;

    private static String PARENT = "MsgData";
    private static String ENTRY = "Transaction";

    XMLStreamReader xmlr = null;
    STAXAppender staxAppender = null;

    public STAXIBRowsetMessage(String filename) {
        try {
            staxAppender = new STAXAppender(filename);
            xmlr = staxAppender.getXMLStreamReader();
            header = staxAppender.getHeader(PARENT);
            footer = staxAppender.getFooter(PARENT);
            xmlr = staxAppender.resetXMLStreamReader();
            header = staxAppender.getHeader(PARENT);
        } catch (Exception ee) {
            ee.printStackTrace();
        }
    }

    public String getHeader() {
        return header;
    }

    public String getFooter() {
        return footer;
    }

    public String getNextTrasaction() {
        try {
            return staxAppender.getEntry(ENTRY, PARENT);
        } catch (Exception ee) {
            ee.printStackTrace();
            return null;
        }
    }

    public static void main(String args[]) {

        STAXIBRowsetMessage srs = new STAXIBRowsetMessage(args[0]);
        String nexttran=null;
        String prevtran=null;
        int count = 0;
        nexttran = srs.getNextTrasaction();
        while (nexttran != null) {
            count++;
            prevtran = nexttran;
            nexttran = srs.getNextTrasaction();
        }
        System.out.println("Last trasaction");
        System.out.println("header:" + srs.getHeader());
        System.out.println("tran:" + prevtran);
        System.out.println("footer:" + srs.getFooter() );
        System.out.println("found " + count + " transaction");

    }
}
```

STAXAppender.java

```

package com.peoplesoft.staxpoc;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.util.Hashtable;

import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamReader;

public class STAXAppender {

    String filename = null;
    String header = null;
    String footer = null;

    XMLInputFactory factory = null;
    XMLStreamReader xmlr = null;
    Hashtable namespaces = new Hashtable();

    public STAXAppender(String filename) throws STAXAppenderException {
        this.filename = filename;
        try {
            factory = XMLInputFactory.newInstance();
            xmlr = factory.createXMLStreamReader(new FileInputStream(filename),
                "UTF-8");
        } catch (XMLStreamException xs) {
            throw new STAXAppenderException(xs.toString());
        } catch (FileNotFoundException fns) {
            System.out.println("Error reading file " + filename);
            throw new STAXAppenderException(fns.toString());
        }
    }

    /*
    start appending when fromTagName is found
    if this is null start from current position
    stop appending
    when toTagName is found (result includes toTagName)
    when untilTagName is found ( return false with out appending any thing found)
    if endofdocument
        return true if append started else return false
    if toTagName is null append till end of document , ignores until tag

    tagType is XMLEvent currently can be only XMLEvent.START_ELEMENT or XMLEvent.END_ELEMENT

    if this is confusing we can write wrapper funtion to have better names
    */
    public boolean append(String fromTagName, String fromURI, int fromTagType,
        String toTagName, String toURI, int toTagType, String untilTagName,
        String untilURI, int untilTagType, StringBuffer appendto)
        throws STAXAppenderException {
        try {
            StringBuffer sb = new StringBuffer("");

            boolean append = false;
            boolean appendtillend = false;

            if (fromTagName == null)
                append = true;

            if (toTagName == null)
                appendtillend = true;

            int evttype = xmlr.getEventType();

            if (evttype == XMLStreamConstants.END_DOCUMENT)
                return false;

            while (xmlr.hasNext()) {

                if (!append) {
                    if (equals(fromTagName, fromURI, fromTagType)) {
                        STAXAppenderUtil.appendEvent(xmlr, sb);
                        append = true;
                    }
                } else {
                    if (appendtillend) {
                        STAXAppenderUtil.appendEvent(xmlr, sb);
                    } else if (equals(toTagName, toURI, toTagType)) {
                        STAXAppenderUtil.appendEvent(xmlr, sb);
                        appendto.append(sb);
                        return true;
                    } else if (equals(untilTagName, untilURI, untilTagType)) {
                        return false;
                    } else {
                        STAXAppenderUtil.appendEvent(xmlr, sb);
                    }
                }
            }
        }
    }
}

```

STAXAppenderUtil.java

```
package com.peoplesoft.staxpoc;

import java.io.FileInputStream;
import java.util.Hashtable;

import javax.xml.stream.XMLInputFactory;
import javax.xml.stream.XMLStreamConstants;
import javax.xml.stream.XMLStreamReader;

public class STAXAppenderUtil {

    public static void appendEvent(XMLStreamReader xmlr, StringBuffer sb) {
        STAXAppenderUtil.appendEvent(xmlr, sb, null);
    }

    public static void appendEvent(XMLStreamReader xmlr, StringBuffer sb,
        Hashtable namespaces) {
        switch (xmlr.getEventType()) {
            case XMLStreamConstants.START_ELEMENT:
                sb.append("<");
                appendName(xmlr, sb);
                appendNameSpaces(xmlr, sb, namespaces);
                appendAttributes(xmlr, sb);
                sb.append(">");
                break;

            case XMLStreamConstants.END_ELEMENT:
                sb.append("</");
                appendName(xmlr, sb);
                // appendAttributes(xmlr, sb);
                sb.append(">");
                break;

            case XMLStreamConstants.SPACE:
            case XMLStreamConstants.CHARACTERS:
                int start = xmlr.getTextStart();
                int length = xmlr.getTextLength();
                sb.append(new String(xmlr.getTextCharacters(), start, length));
                break;

            case XMLStreamConstants.PROCESSING_INSTRUCTION:
                sb.append("<?");
                if (xmlr.hasText())
                    sb.append(xmlr.getText());
                sb.append(">");
                break;

            case XMLStreamConstants.CDATA:
                sb.append("<![CDATA[");
                if (xmlr.hasText())
                    sb.append(xmlr.getText());
                sb.append("]]>");
                break;

            case XMLStreamConstants.COMMENT:
                sb.append("<!--");
                if (xmlr.hasText())
                    sb.append(xmlr.getText());
                sb.append("-->");
                break;

            case XMLStreamConstants.ENTITY_REFERENCE:
                sb.append(xmlr.getLocalName() + "=");
                if (xmlr.hasText())
                    sb.append("[ " + xmlr.getText() + " ]");
                break;

            case XMLStreamConstants.START_DOCUMENT:
                sb.append("<?xml");
                sb.append(" version=\"" + xmlr.getVersion() + "\"");
                if (xmlr.getCharacterEncodingScheme() != null)
                    sb.append(" encoding=\"" + xmlr.getCharacterEncodingScheme()
                        + "\"");
                if (xmlr.isStandalone())
                    sb.append(" standalone='yes'");
                //else
                // sb.append(" standalone='no'");
                sb.append(">\n");
                break;
        }
    }

    private static void appendName(XMLStreamReader xmlr, StringBuffer sb) {
        if (xmlr.hasName()) {
            String prefix = xmlr.getPrefix();
            String uri = xmlr.getNamespaceURI();
            String localName = xmlr.getLocalName();
            appendName(prefix, uri, localName, sb);
        }
    }
}
```

STAXAppenderException.java

```
package com.peoplesoft.staxpoc;

public class STAXAppenderException extends Exception {
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public STAXAppenderException(String msg) {
        super(msg);
    }
}
```
