XPath Evaluation

This example shows the usage of Xpath and Velocity. It permits the usage of XPath in foreach directives.

This sample has a dependency to Saxon. Libraries used:

- saxon8.jar
- saxon8-dom.jar
- velocity-1.5.jar
- commons-collections-3.1.jar
- commons-lang-2.1.jar

Run the test.Sample class.

File: *test.v*

```
package ${packageName};

public class My${xpath.run("xs:string(//package/class/@name)",$mydoc)}
{
    public My${xpath.run("xs:string(//package/class/@name)",$mydoc)}()
    {
#foreach($field in $xpath.iterator("//package/class/field",$mydoc))
        this.${xpath.run("xs:string(//field/@name)",$field)} = "1";
#end
    }
}
```

File: *test.Sample*

```
package test;

import java.io.StringWriter;

import org.apache.velocity.Template;
import org.apache.velocity.VelocityContext;
import org.apache.velocity.app.VelocityEngine;

public class Sample
{
    public String generate(String templateFile,String rawDoc) throws Exception
    {
        VelocityEngine engine = new VelocityEngine();
        Template t = engine.getTemplate(templateFile);
        VelocityContext context = new VelocityContext();
        context.put("mydoc", rawDoc);
        context.put("xpath", new XPathHandler());
        context.put("packageName", "org.test");
        StringWriter wr = new StringWriter();
        t.merge(context, wr);
        return wr.getBuffer().toString();
    }

    /**
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception
    {
        String templateFile = "test.v";
        String rawDoc = "<package><class name=\"Director\"><field name=\"test1\"/></class></package>";
        System.out.println(new Sample().generate(templateFile, rawDoc));
    }

}
```

File: *test.XPathHandler*

```
package test;

import java.io.StringReader;
import java.io.StringWriter;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Properties;

import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

import net.sf.saxon.Configuration;
import net.sf.saxon.om.DocumentInfo;
import net.sf.saxon.query.DynamicQueryContext;
import net.sf.saxon.query.StaticQueryContext;
import net.sf.saxon.query.XQueryExpression;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;

public class XPathHandler
{

    /**
     * Run a Xpath query using Saxon
     * @param xpath
     * @param doc either an instance of {@link Element} or a raw xml {@link String}.
     * @return
     */
    public String run(String xpath, Object doc)
    {
        try
        {
            Configuration config = new Configuration();
            StaticQueryContext staticContext = new StaticQueryContext(config);
            StringReader reader = new StringReader(xpath);
            XQueryExpression exp = staticContext.compileQuery(reader);
            DynamicQueryContext dynamicContext = new DynamicQueryContext(config);

            if( doc instanceof String )
            {
                DocumentInfo d = staticContext.buildDocument(new StreamSource(new StringReader((String)doc))).
getDocumentRoot();
                dynamicContext.setContextNode(d);
            }
            else if( doc instanceof Element)
            {
                DOMSource domSource = new DOMSource((Element)doc);
                StringWriter writer = new StringWriter();
                StreamResult result = new StreamResult(writer);
                TransformerFactory tf = TransformerFactory.newInstance();
                Transformer transformer = tf.newTransformer();
                Properties props = new Properties();
                props.setProperty(OutputKeys.METHOD, "xml");
                props.setProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");

                transformer.setOutputProperties(props);
                transformer.transform(domSource, result);

                DocumentInfo d = staticContext.buildDocument(new StreamSource(new StringReader(writer.
toString()))).getDocumentRoot();
                dynamicContext.setContextNode(d);
            }
            else
```

```
            {
                return null;
            }
            Properties props = new Properties();
            props.setProperty(OutputKeys.METHOD, "text");
            props.setProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
            props.setProperty(OutputKeys.STANDALONE, "yes");
            StringWriter wr = new StringWriter();
            exp.run(dynamicContext,new StreamResult(wr), props);
            return wr.getBuffer().toString();
        }
        catch (Exception e)
        {
            throw new RuntimeException(e);
        }
    }

    /**
     * Run a xpath and return a collection of elements
     * @param xpath
     * @param rawdoc
     * @return Collection of {@link Element} elements
     */
    public Collection iterator(String xpath, String rawdoc)
    {
        try
        {
            Configuration config = new Configuration();
            StaticQueryContext staticContext = new StaticQueryContext(config);
            StringReader reader = new StringReader(xpath);
            XQueryExpression exp = staticContext.compileQuery(reader);
            DynamicQueryContext dynamicContext = new DynamicQueryContext(config);

            DocumentInfo doc = staticContext.buildDocument(new StreamSource(new StringReader(rawdoc))).
getDocumentRoot();
            Properties props = new Properties();
            props.setProperty(OutputKeys.METHOD, "xml");
            props.setProperty(OutputKeys.OMIT_XML_DECLARATION, "yes");
            dynamicContext.setContextNode(doc);
            StringWriter wr = new StringWriter();

            exp.run(dynamicContext,new StreamResult(wr), props);

            String xml ="<root>"+wr.getBuffer().toString()+"</root>"; //wraps in a root element, for getting a
list of elements later
            DocumentBuilderFactory dom = javax.xml.parsers.DocumentBuilderFactory.newInstance();
            Document document = dom.newDocumentBuilder().parse(new InputSource(new StringReader(xml)));

            return new NodeListWrapper(document.getDocumentElement().getChildNodes());

        }
        catch (Throwable e)
        {
            throw new RuntimeException(e);
        }
    }

    /**
     * Wraps a NodeList within an ArrayList. only accepts {@link Element} as elements
     */
    public static class NodeListWrapper extends ArrayList
    {
        public NodeListWrapper(NodeList si)
        {
            try
            {
                for(int i=0; i<si.getLength(); i++)
                {
                    if( si.item(i) instanceof Element)
                    {
                        add(si.item(i));
```

```
                    }
                }
            }
            catch (Exception e)
            {
                //ignore
            }
        }
    }
}
```