JAXP Document Validation

Here's a nice class to validate XML Documents in Java:

```
package co.wlv.xml.validation;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;

import javax.xml.XMLConstants;
import javax.xml.transform.dom.DOMSource;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.w3c.dom.Document;
import org.w3c.dom.ls.LSResourceResolver;
import org.xml.sax.ErrorHandler;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class XsdValidator {
    private static Logger logger = LoggerFactory.getLogger(Spike.class);

    private URL schemaUrl;
    private Schema schema;
    private LSResourceResolver resourceResolver;

    public XsdValidator(URL schemaUrl) throws SAXException {
        setSchemaUrl(schemaUrl);
        init();
    }

    public XsdValidator(URL schemaUrl, LSResourceResolver resourceResolver) throws SAXExcep
        setSchemaUrl(schemaUrl);
        setResourceResolver(resourceResolver);
        init();
    }

    public List<SAXParseException> validateDocumentWithXsd(Document doc) {
        XsdErrorHandler xsdErrorHandler = new XsdErrorHandler();
        Validator validator = getSchema().newValidator();
        validator.setErrorHandler(xsdErrorHandler);

        DOMSource source = new DOMSource(doc);
        try {
            validator.validate(source);
        }
        catch (SAXException e) {
            logger.error("ERROR: validateDocumentWithXsd", e);
        }
        catch (IOException e) {
            logger.error("ERROR: validateDocumentWithXsd", e);
        }
        return xsdErrorHandler.getSaxParseExceptions();
    }

    public URL getSchemaUrl() {
        return schemaUrl;
    }

    public void setSchemaUrl(URL schemaUrl) {
        this.schemaUrl = schemaUrl;
    }
```

```
    public LSResourceResolver getResourceResolver() {
        return resourceResolver;
    }

    public void setResourceResolver(LSResourceResolver resourceResolver) {
        this.resourceResolver = resourceResolver;
    }

    private void init() throws SAXException {
        SchemaFactory sf = SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_NS_URI);
        if (resourceResolver != null)
            sf.setResourceResolver(resourceResolver);
        setSchema(sf.newSchema(getSchemaUrl()));
    }

    private Schema getSchema() {
        return schema;
    }

    private void setSchema(Schema schema) {
        this.schema = schema;
    }

    private class XsdErrorHandler implements ErrorHandler {
        private List<SAXParseException> saxParseExceptions = new ArrayList<SAXParseExceptio

        @Override
        public void warning(SAXParseException e) throws SAXException {
            if (logger.isDebugEnabled())
                logger.debug(String.format("WARNING: %s", e.getMessage()));
            saxParseExceptions.add(e);
        }

        @Override
        public void fatalError(SAXParseException e) throws SAXException {
            if (logger.isDebugEnabled())
                logger.debug(String.format("FATAL: %s", e.getMessage()));
            saxParseExceptions.add(e);
        }

        @Override
        public void error(SAXParseException e) throws SAXException {
            if (logger.isDebugEnabled())
                logger.debug(String.format("ERROR: %s", e.getMessage()));
            saxParseExceptions.add(e);
        }

        public List<SAXParseException> getSaxParseExceptions() {
            return saxParseExceptions;
        }
    }
}
```

And a test case...

```


package co.wlv.spike.SpikeSchematron;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.Reader;
import java.net.URL;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.regex.Pattern;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.TransformerException;
```

```
import org.apache.log4j.xml.DOMConfigurator;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.w3c.dom.Document;
import org.w3c.dom.ls.LSInput;
import org.w3c.dom.ls.LSResourceResolver;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

public class Spike {
    private static Logger logger = LoggerFactory.getLogger(Spike.class);
    private static File APIP_BASE = new File("etc/APIPv1p0_FinalRelease_20121017/Core_Level
    private static File APIP_ITEM_XSD = new File(APIP_BASE, "apipv1p0_qtiitemv2p1_v1p0.xsd"

    private static File APIP_ITEM_XML = new File("etc/samples/Item/VF656330.qti.xml");

    /**
     * @param args
     * @throws TransformerException
     * @throws ParserConfigurationException
     * @throws IOException
     * @throws SAXException
     */
    public static void main(String[] args) throws TransformerException, ParserConfiguration
        DOMConfigurator.configure("etc/log4j.xml");

        Spike spike = new Spike();
        spike.doit();

        logger.debug("DONE!");
    }

    private void doit() throws ParserConfigurationException, SAXException, IOException {
        URL schemaUrl = APIP_ITEM_XSD.toURI().toURL();

        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        dbf.setNamespaceAware(true);
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document doc = db.parse(APIP_ITEM_XML);

        XsdValidator xsdValidator = new XsdValidator(schemaUrl, new ApipLSResourceResolver(

        for (int i = 0; i < 1000; i++) {
            List<SAXParseException> saxParseExceptions = xsdValidator.validateDocumentWithX
            logger.debug(i + ") COUNT: " + saxParseExceptions.size());
        }
    }

    private class ApipLSResourceResolver implements LSResourceResolver {
        private Set<String> resolvedResources = new HashSet<String>();
        private Pattern apipPattern = Pattern.compile("^http://www.imsglobal.org/profile/ap

        @Override
        public LSInput resolveResource(String type, String namespaceURI, String publicId, S
            if (logger.isDebugEnabled()) {
                logger.debug("============================================== resolveResource
                logger.debug(String.format("NS_URI: %s; ", namespaceURI));
                logger.debug(String.format("PUB_ID: %s; ", publicId));
                logger.debug(String.format("SYS_ID: %s;", systemId));
                logger.debug(String.format("BASE_URI: %s", baseURI));
            }
            LSInput lsInput = null;
            if (systemId != null && !resolvedResources.contains(systemId)) {
                try {
                    String name = null;
                    if (systemId.equals("http://www.imsglobal.org/xsd/w3/2001/xml.xsd"))
                        name = "etc/xsd/xml.xsd";
                    else if (systemId.equals("http://www.imsglobal.org/xsd/w3/2001/XInclude
                        name = "etc/xsd/XInclude.xsd";
                    else if (systemId.equals("http://www.w3.org/Math/XMLSchema/mathml2/math
                        name = "etc/xsd/mathml2/mathml2.xsd";
                    else if (apipPattern.matcher(systemId).matches())
                        name = String.format("%s/%s", APIP_BASE, systemId.replaceAll("^.*/"

                    if (name != null) {
                        File file = new File(name);
```

```
                    if (logger.isDebugEnabled())
                        logger.debug("FILE: {}; EXISTS: {}; URI: {}", file, file.exists
                    if (file.exists()) {
                        lsInput = new LSInputImpl(new FileInputStream(file));
                        lsInput.setSystemId(file.toURI().toString());
                    }
                }
                resolvedResources.add(systemId);
            }
            catch (FileNotFoundException e) {
                logger.error("Error resolving resource.", e);
            }
        }
        return lsInput;
    }
}

private class LSInputImpl implements LSInput {

    private String baseUri, encoding, publicId, systemId;
    private InputStream byteStream;

    public LSInputImpl(InputStream byteStream) {
        this.byteStream = byteStream;
    }

    @Override
    public String getBaseURI() {
        return baseUri;
    }

    @Override
    public InputStream getByteStream() {
        return byteStream;
    }

    @Override
    public boolean getCertifiedText() {
        // TODO Auto-generated method stub
        return false;
    }

    @Override
    public Reader getCharacterStream() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String getEncoding() {
        return encoding;
    }

    @Override
    public String getPublicId() {
        return publicId;
    }

    @Override
    public String getStringData() {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public String getSystemId() {
        return systemId;
    }

    @Override
    public void setBaseURI(String baseURI) {
        baseUri = baseURI;
    }

    @Override
    public void setByteStream(InputStream byteStream) {
```

```
            this.byteStream = byteStream;
        }

        @Override
        public void setCertifiedText(boolean certifiedText) {
            // TODO Auto-generated method stub
        }

        @Override
        public void setCharacterStream(Reader characterStream) {
            // TODO Auto-generated method stub
        }

        @Override
        public void setEncoding(String encoding) {
            this.encoding = encoding;
        }

        @Override
        public void setPublicId(String publicId) {
            this.publicId = publicId;
        }

        @Override
        public void setStringData(String stringData) {
            // TODO Auto-generated method stub
        }

        @Override
        public void setSystemId(String systemId) {
            this.systemId = systemId;
        }
    }
}
```

Published by Wyatt
Mon Aug 19 23:37:10 EDT 2013
http://blog.wylovan.com/pebble/wyatt/2013/08/19/1376969830289.html