XML to Lucene to SRW

(Work Area B.2 - B.4)

Serhiy Polyakov

February 15, 2007

Table of Contents

1. Introduction

The data workflow that has been chosen for implementation can be described as DAMS -> XML -> Lucene -> SRU/SRW -> Texas Heritage Online.

The document *Parsing Records from TSLAC ARIS Databases into XML: Notes* describes the process and includes code for exporting data form MySQL database into XML records in native and simple DC schemas.

This document describes the procedures of parsing XML records into Lucene index and making those records available via SRW.

2. Parsing XML records into to Lucene index

2.1. Installing Lucene

Lucene is a free and open source information retrieval API, originally implemented in Java. Java runtime environment (or SDK) should be installed on the server in order to run the code. Linux Fedora Core 4 server has been used in this implementation.

Note: SRW/SRU Lucene interface (will be described in the next section) developed by Ralph LeVan requires version 1.5+ of Java.

Installing Java SDK
Download:
jdk-1_5_0_10linux-i586.rpm.bin

chmod +x jdk-1_5_0_10linux-i586.rpm.bin

Run it:
./jdk-1_5_0_10linux-i586.rpm.bin
It will install into /usr/java/jdk1.5.0_10

Add to /etc/profile
```
JAVA_HOME="/usr/java/jdk1.5.0_10"
export JAVA_HOME
```

Also add (for tomcat55)
```
JRE_HOME="/usr/java/jdk1.5.0_10"
export JRE_HOME
```

Note: It is necessary to tell Fedora that this version of Java is preferred interpreter rather than the Open Source version that's installed by default:
# /usr/sbin/alternatives --install /usr/bin/java java /usr/java/jdk1.5.0_10/bin/java 1510

Installing Lucene

The Java version on Lucene is just another JAR file.

Note: Book examples and some sample codes from various sources cannot be compiled from command line because they contain methods removed in Lucene 2.0. These code samples work with Lucene 1.9.

Download
lucene-1.9-final.zip

Extract and copy lucene-core-1.9-final.jar into
/usr/java/jdk1.5.0_10/lib

Update CLASSPATH variable:
```
# export CLASSPATH=/usr/java/jdk1.5.0_10/lib/lucene-core-1.9-final.jar
```
Close shell and reopen it and the new variable will be read.

2.2. XML to Lucene parser

The parser for converting/indexing data from XML documents into Lucene index has been developed using Digester. Digester is an open source projects from the Apache Foundation. It offers high-level interface for mapping XML documents to Java objects. Digester requires a few additional Java libraries. Digester's home page provides a short list of the libraries. Digester and two additional libraries (Logging 1.1.x and BeanUtils 1.7) have been installed.

The parser comprise java program that reads XML documents from directory, indexes each file, and updates Lucene index. Parser is specific for the XML schema and has been developed for the schema presented below. This schema is used to store records from ARIS Service Record database.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="service_record" type="service_recordType"/>

    <xsd:complexType name="service_recordType">

        <xsd:sequence>

            <xsd:element name="id" type="xsd:int"/>
            <xsd:element name="name" type="xsd:string"/>
            <xsd:element name="call_number" type="xsd:string"/>
            <xsd:element name="organization_code" type="xsd:string"/>
            <xsd:element name="organization_title" type="xsd:string"/>
            <xsd:element name="image_URL" type="xsd:string"/>

        </xsd:sequence>

    </xsd:complexType>

</xsd:schema>
```

However, it is possible to develop universal parser that would utilize XML schemas for the input documents.

The code for the parser in presented in Appendix A.

Running the code

XMLtoLuceneServiceRecord.java should be compiled:
```
#javac  XMLtoLuceneServiceRecord.java
```

Then it can be run from command line:
```
#java  XMLtoLuceneServiceRecord
```

XML documents will be red from current directory and index will be written to the index supdirectory of the temporary directory as defined by the function:
System.getProperty("java.io.tmpdir", "tmp")

To run the code against multiple files shell script can be used under Linux or bat file under Windows.

200 XML documents (derivatives of ARIS Service Records database) have been indexed into Lucene. Lucene index consists of set of files in one directory.

2.3. SWRLucene

Ralph Levan at OCLC has developed SRWLucen service that provides SRW interface to Lucene index. The service runs under Tomcat, a Servlet and Java Server Pages container.

Installing Tomcat
Note: SRWLucene requires Java 1.5+. Fedora built-in Tomcat hard wired to use open source implementation of java (1.4.2) which comes with Fedora. Standalone JDK 1.5+ and Tomcat should be installed.

Tomcat service has been installed on test server on port 8080:

Stop tomcat5 (came with Fedora)

Before installing tomcat55 needed modules (dependencies) have been installed with yum:
jakarta-commons-daemon
xerces-j2
struts

Download 11 tomcat55 rpms from
http://www.jpackage.org/browser/rpm.php?jppversion=1.6&id=3553

```
rpm -ihh tomcat55*
```

Start
```
service tomcat55 start
OK
```

Test that Tomcat is working by typing http://localcost:8080 into the browser. Tomcat welcome page should come.

The Tomcat webbapps directory is:
/var/lib/tomcat55/webapps

There is no need to install the connector between Apache web Server and Tomcat because Tomcat can be functioning in standalone mode. Port number (8080) should be specified in the URL in this case.

Installing SRWLucene service

SRWLucene service has been deployed and configured for the test Lucene index containing 200 records from ARIS database. This index can be located in any directory and path should be configured as described below.

SRW service developed by Ralph LeVan can be downloaded from:

http://staff.oclc.org/~levan/SRWLucene.war

This .war file should be copied into tomcat webbapps directory:
/var/lib/tomcat55/webapps

Tomcat will unpack it. Inside the SRWLucene/WEB-INF/classes directory there is the SRWServer.props file. Entry for Lucene database (index) should be added to this file.

Here is SRWServer.props file configured for this example:

```
#db.<databaseName>.class=org.osuosl.srw.lucene.SRWLuceneDatabase.class
db.THDILuceneDemoDB.class=ORG.oclc.os.SRW.Lucene.SRWLuceneDatabase

#db.<databaseName>.home=<full pathname to your Lucene database directory> (optional).
#If omitted, all files needed to initialize the database must be in the services CLASSPATH.
#Typically, this means putting them in the webapps/SRW/WEB-INF/classes directory.
db.THDILuceneDemoDB.home=/home/serhiyp/luceneindex/

#db.<databaseName>.configuration=SRWDatabase.props (optional).
#If the database home directory was specified and that directory or one of its subdirectories
#contains the Lucene index, then the SRWLuceneDatabase will be able to provide an acceptable
#minimal configuration automatically.
db.THDILuceneDemoDB.configuration=SRWDatabase.props
```

Also, there is a database configuration file located in the Lucene database (index) derectory. It will allow use of stylesheet based user interface.

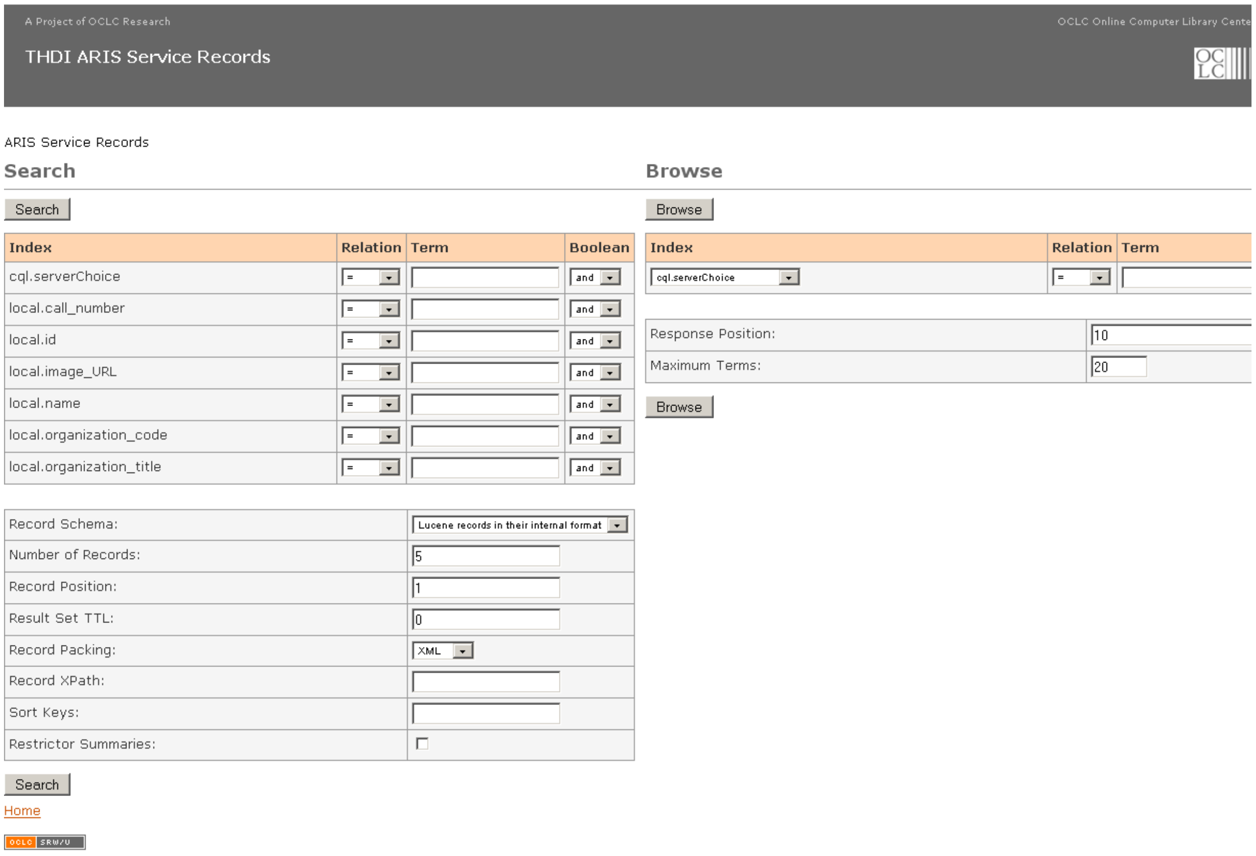
Here is SRWServer.props file configured for this example:

```
####General Database Information

#The name of the database (recommended).
databaseInfo.title=THDI Lucene Demo

#A brief description of the database (optional).
databaseInfo.description=An index of some text files

#The author/creator of the database (optional).
databaseInfo.author=Serhiy Polyakov

#A person to contact with questions/problems (recommended).
databaseInfo.contact=Serhiy

#Any usage restrictions (optional).
databaseInfo.restrictions=No restrictions


####Information about the Explain record

#The date the record was last modified (optional).
metaInfo.dateModified=2006-12-01

#If the record was collected from another site, the URL of the original record (optional).
metaInfo.aggregatedFrom=Lucene in Action Samples

#If the record was collected, the date the record was collected (optional).
metaInfo.dateAggregated=2006-12-10


####Default configuration values

#The maximum number of records that can be returned in a response. Default is 20 (optional).
configInfo.maximumRecords=15

#The number of records to return in a response if not specified in the request.
```

```
#Default is 10 (optional).
configInfo.numberOfRecords=5

#The number of seconds that query results should be kept, if not specified in the request.
#Default is 300 (optional).
configInfo.resultSetTTL=200
```

Once a database (index) is added Tomcat is restarted, it will be accessible at http://<host>:<port>/SRWLucene/search/<databaseName>. That will cause an Explain record for the database to be returned. If the server is pointed at a configuration file with stylesheets specified, then search interface for the database will be available and the explain record will be visible when browse commend is selected to display the document source.

Accessing the records via SRW
The service can be accessed trough web interface:
http://spmachine.lis.unt.edu:8080/SRWLucene/search/THDI_ARIS_service_records
This will show Explain record for the database. Search and Index Browse functionality are available.



A Project of OCLC Research

OCLC Online Computer Library Cente

THDI ARIS Service Records

ARIS Service Records

Search

Index	Relation	Term	Boolean
cql.serverChoice	=		and
local.call_number	=		and
local.id	=		and
local.image_URL	=		and
local.name	=		and
local.organization_code	=		and
local.organization_title	=		and

Record Schema:	Lucene records in their internal format
Number of Records:	5
Record Position:	1
Result Set TTL:	0
Record Packing:	XML
Record XPath:	
Sort Keys:	
Restrictor Summaries:	☐

Search

Home

Browse

Index	Relation	Term
cql.serverChoice	=	

Response Position:	10
Maximum Terms:	20

Browse

To use the service with any kind of SOAP client the following pointer to Web Service Definition can be used:
http://spmachine.lis.unt.edu:8080/SRWLucene/search/THDI_ARIS_service_records?wsdl

Appendix A: XML to Lucene parser for ARIS Service Records database

```
import org.apache.commons.digester.Digester;
import org.xml.sax.SAXException;
import org.apache.lucene.index.IndexWriter;
import org.apache.lucene.analysis.Analyzer;
import org.apache.lucene.analysis.WhitespaceAnalyzer;
import org.apache.lucene.document.Document;
import org.apache.lucene.document.Field;

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


/**
 * Parses the contents of service record XML file and indexes all
 * contact entries found in it.  The name of the file to parse must be
 * specified as the first command line argument.
 */
public class XMLtoLuceneServiceRecord
{
    private static IndexWriter writer;

    /**
     * Adds the contact to the index.
     *
     * @param contact the <code>Contact</code> to add to the index
     */
    public void addContact(Contact contact) throws IOException
    {
        System.out.println("Adding " + contact.getId());
        Document contactDocument  = new Document();
        contactDocument.add(Field.Text("name", contact.getName()));
        contactDocument.add(Field.Text("id", contact.getId()));
        contactDocument.add(Field.Text("call_number", contact.getCall_number()));
        contactDocument.add(Field.Text("organization_code", contact.getOrganization_code()));
        contactDocument.add(Field.Text("organization_title", contact.getOrganization_title()));
        contactDocument.add(Field.Text("image_URL", contact.getImage_URL()));

        writer.addDocument(contactDocument);
    }

    /**
     * Created an index to add contacts to, configures Digester rules and
     * actions, parses the XML file specified as the first argument.
     *
     * @param args command line arguments
     */
    public static void main(String[] args) throws IOException, SAXException
    {
        String indexDir =
            System.getProperty("java.io.tmpdir", "tmp") +
           System.getProperty("file.separator") + "index";
        Analyzer analyzer = new WhitespaceAnalyzer();
        //boolean createFlag = true;
        boolean createFlag = false;

        // IndexWriter to use for adding contacts to the index
        writer = new IndexWriter(indexDir, analyzer, createFlag);
        //writer = new IndexWriter(indexDir, analyzer, false);


        // instantiate Digester and disable XML validation
        Digester digester = new Digester();
        digester.setValidating(false);

           // instantiate XMLtoLuceneServiceRecord class
        digester.addObjectCreate("service_record", XMLtoLuceneServiceRecord.class );
        // instantiate Contact class
        digester.addObjectCreate("service_record", Contact.class );



        // set different properties of Contact instance using specified methods
        digester.addCallMethod("service_record/name",       "setName", 0);
        digester.addCallMethod("service_record/id",       "setId", 0);
        digester.addCallMethod("service_record/call_number",       "setCall_number", 0);
        digester.addCallMethod("service_record/organization_code",       "setOrganization_code", 0);
        digester.addCallMethod("service_record/organization_title",       "setOrganization_title", 0);
        digester.addCallMethod("service_record/image_URL",       "setImage_URL", 0);


        // call 'addContact' method when the next 'address-book/contact' pattern is seen
        digester.addSetNext("service_record",               "addContact" );


        // now that rules and actions are configured, start the parsing process
        //SP XMLtoLuceneServiceRecord X2L = (XMLtoLuceneServiceRecord) digester.parse(new
File(args[0]));
              XMLtoLuceneServiceRecord X2L = (XMLtoLuceneServiceRecord) digester.parse(new
File(args[0]));

        // optimize and close the index
        writer.optimize();
        writer.close();
    }

    /**
     * JavaBean class that holds properties of each Contact entry.
     * It is important that this class be public and static, in order for
     * Digester to be able to instantiate it.
     */
    public static class Contact
    {
              private String name;
        private String id;
        private String call_number;
        private String organization_code;
        private String organization_title;
        private String image_URL;

        public void setName(String newName)
        {
           name = newName;
        }
        public String getName()
        {
           return name;
        }

        public void setId(String newId)
        {
           id = newId;
        }
        public String getId()
        {
           return id;
        }

        public void setCall_number(String newCall_number)
        {
           call_number = newCall_number;
        }
        public String getCall_number()
        {
           return call_number;
        }

        public void setOrganization_code(String newOrganization_code)
        {
           organization_code = newOrganization_code;
        }
        public String getOrganization_code()
        {
           return organization_code;
        }

              public void setOrganization_title(String newOrganization_title)
        {
           organization_title = newOrganization_title;
        }
        public String getOrganization_title()
        {
           return organization_title;
        }

              public void setImage_URL(String newImage_URL)
        {
           image_URL = newImage_URL;
        }
        public String getImage_URL()
        {
           return image_URL;
        }
```

```
    }
}
```