XSLTranslator

Overview

This class enables translation of XML into RDF using a XSL File. The main intent of this class is that the user will define the translation from the target data source's schema into XML/RDF that follows the VIVO ontology's schema. This means that without recompiling users can create Harvesting methods from various XML sources using different XSL files for each.

The author of the XSL file needs to have knowledge of the incoming data and the ontology of the vivo being harvested into. If there is an ontology specialist available they should investigate the post translated RDF/XML to be certain that the data is being placed in the proper format.

This class uses Saxon's implementation of an xsl translator as it translates between RecordHandlers.

Parameters

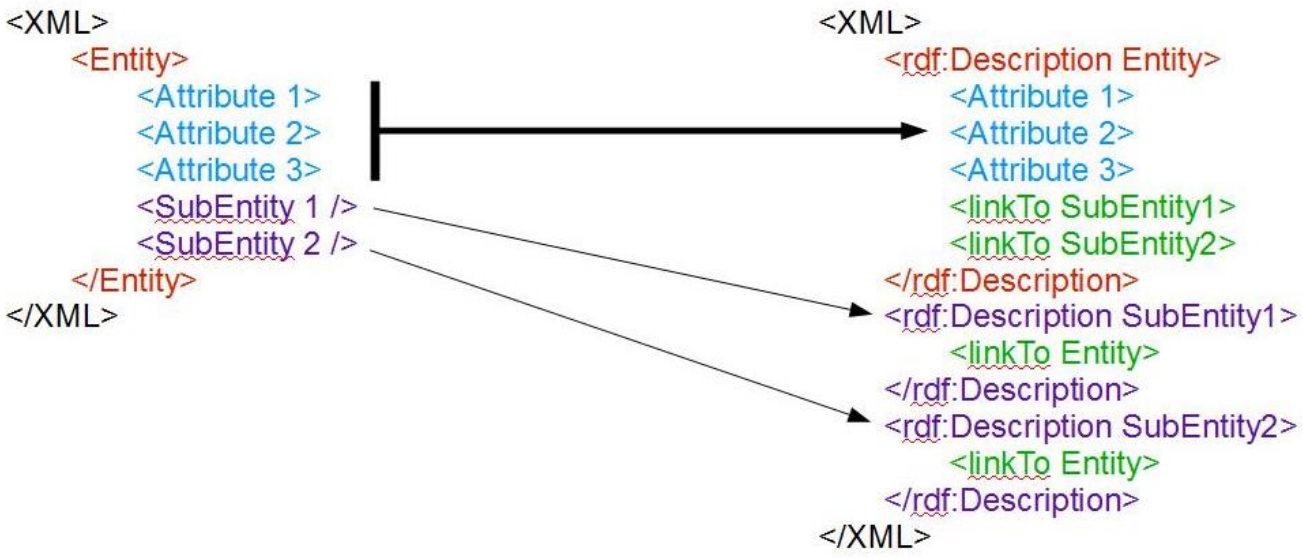
Short Option	Long Option	Parameter Value Map	Description	Required
i	input	CONFIG_FILE	config file for input record handler	true
I	inputOverride	RH_PARAM	override the RH_PARAM of input record handler using VALUE	false
o	output	CONFIG_FILE	config file for output record handler	true
O	outputOverride	RH_PARAM	override the RH_PARAM of output record handler using VALUE	false
x	xslFile	XSL_FILE	the stylesheet to use for translating records from input to output	true
f	force		force translation of all input records, even if previously processed	false

Variables

Name	Type	Visibility	Description
translationString	String	private	The translation xsl is the map that will reconstruct our input stream's document into the appropriate format
inStore	RecordHandler	protected	record handler for incoming records
outStore	RecordHandler	protected	record handler for storing records

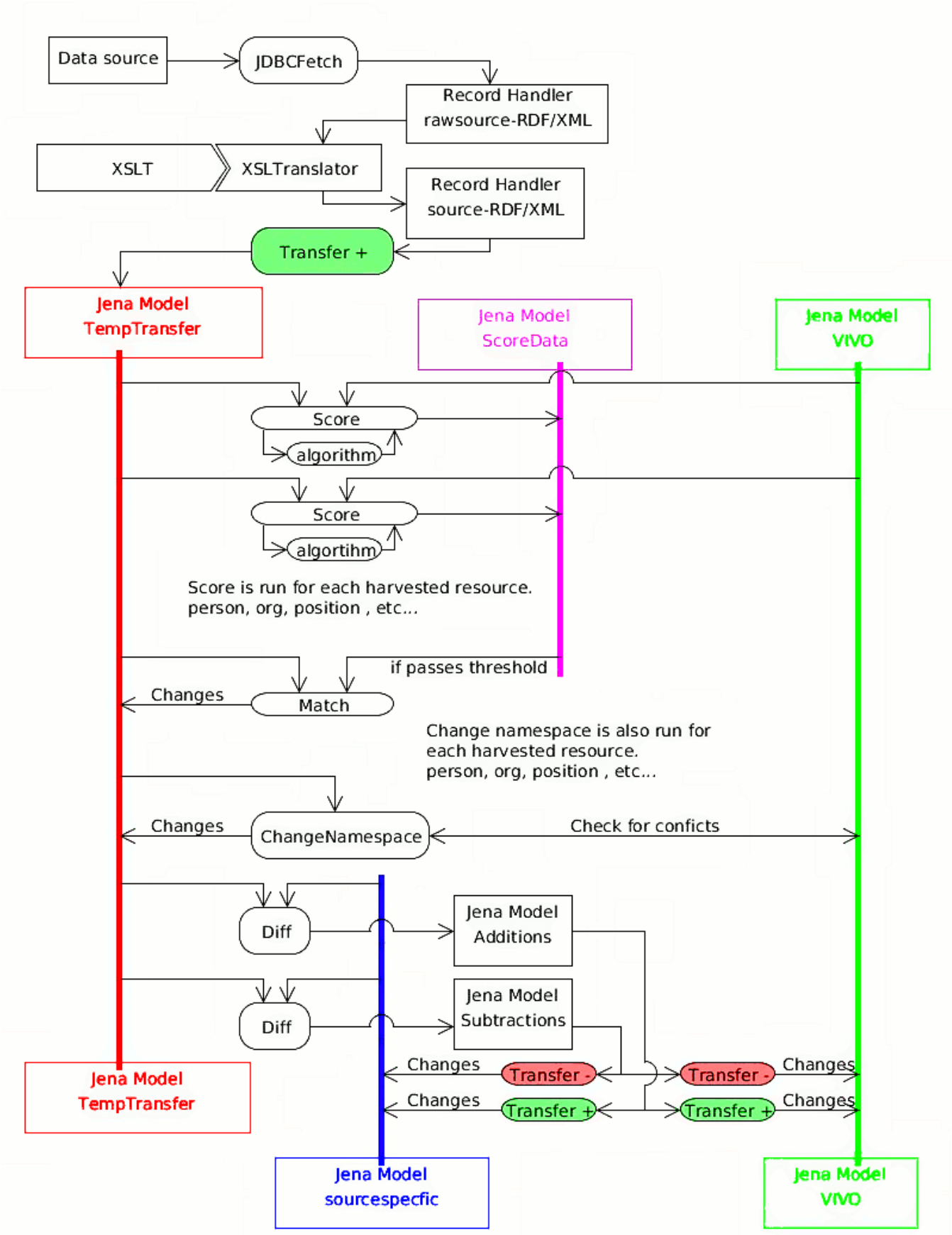
Translation File

For each Translation File there will be a Detail Design of how the XSLT is designed to translate the original schema into the VIVO ontology's schema. The main change, aside from the VIVO ontology, is the format of XML/RDF. Each entity, including sub entities, is flattened. Links exist within each entity to the other entities its related to.



Flowchart

The XSLTranslator is an integral part of the standard JDBC fetch.


Data source
JDBCFetch
Record Handler
rawsource-RDF/XML
XSLT
XSLTranslator
Record Handler
source-RDF/XML
Transfer +
Jena Model
TempTransfer
Jena Model
ScoreData
Jena Model
VIVO
Score
algorithm
Score
algortihm
Score is run for each harvested resource.
person, org, position , etc...
if passes threshold
Changes
Match
Change namespace is also run for
each harvested resource.
person, org, position , etc...
Changes
ChangeNamespace
Check for conficts
Diff
Jena Model
Additions
Diff
Jena Model
Subtractions
Changes
Transfer -
Transfer -
Changes
Changes
Transfer +
Transfer +
Changes
Jena Model
TempTransfer
Jena Model
sourcespecfic
Jena Model
VIVO

Tutorials on XSLT

- XSLT Tutorial - http://www.w3schools.com/xsl/xsl_languages.asp
- Xpath Tutorial - http://www.w3schools.com/xpath/default.asp
- XSL-FO Tutorial - http://www.w3schools.com/xslfo/default.asp

Example XSLT Designs

- PubMed to VIVO XSL Information
- hCard Microformat to VIVO XSL Information
- hGrant Microformat to VIVO XSL Information
- hResume Microformat to VIVO XSL Information
- hCalendar Microformat to VIVO XSL Information
- rel-tag Microformat to VIVO XSL Information
- citation Microformat to VIVO XSL Information
- OAI__DC to VIVO Information

Execution

Define Alias

XSLTranslator="java $OPTS ~~-Xms$MIN_MEM -Xmx$MAX_MEM -Dharvester.task=$HARVESTER_TASK -Dprocess.task=XSLTranslator -cp bin/harvester~~ $VERSION.jar:bin/dependency/* org.vivoweb.harvester.translate.XSLTranslator"

Invocation

$XSLTranslator -i config/recordHandlers/PeopleSoft-Merge.xml -o config/recordHandlers/PeopleSoft-RDF.xml -x config/datamaps/PeopleSoftToVivo.xsl

Methods

setTranslation

Loads the file into memory as an attempt to speed up translations.

1. create ByteArrayOutputStream
2. copy the InputStream into ByteOutputStream
3. create string from ByteOutputStream

execute

The execute method serves as a wrapper around the actual xml translation method.

1. set translated and passed counters to zero
2. loop through all the records in inStore
    a. create a ByteArrayOutputStream
    b. execute xmlTranslate with:
        i. a stream based off the loop's record
        ii. a stream based off translationString
        iii. the new ByteArrayOutputStream
    c. add translated record to outStore.

xmlTranslate

This Method uses the Javax xml transform factory to apply the xsl

1. create javax.xml.transform.stream.StreamResult from the OutputStream
2. create javax.xml.transform.Source from the InputStream
3. create javax.xml.transform.Source from the translationStream (of type InputStream)
4. use javax.xml.transform.TransformerFactory to apply the transform on the input to produce the output.