

XSLT Ingest Example: Appendix A

Appendix A

[Start](#) [Previous](#) [Next](#)

The following table shows the directory hierarchy of the example.

example/: acc/ bin/ rdf/ readme/ source/ xmls/ xslt/	0
example/acc: Org0.xml Per0.xml	1
example/bin: nuno*	2
example/rdf: ED.rdf NewOrgs.rdf NewPers.rdf	3
example/readme: CommandSequence.txt	4
example/source: EduRS.xml	5
example/xmls: ED0.xml ED2.xml EX-URP-UNOs.xml NewPers.xml ED1.xml EX-URO-UNOs.xml NewOrgs.xml	6
example/xslt: auxfuncs.xml fillURPs.xml mkAccOrgRdf.xml <u>saxon9he.jar</u> countUROs.xml gather.xml mkAccPeopleRdf.xml countURPs.xml makeUROs.xml mkEduRdf.xml fillUROs.xml makeURPs.xml	7

Directory Layout - Figure 16

- [F16H0] This is a list of the **example** subdirectories.
- [F16H1] This is where the accumulator files are found. **Per0.xml** for people and **Org0.xml** for organizations. In the case of this example, there are only a few entries in each of these files. In [vivo.cornell.edu](#) there are tens of thousands of people and thousands of organizations. These files are constructed by SPARQL query and XSLT. Since this is really a sideshow, it won't be covered in detail here (see [Appendix E](#) for the SPARQL query and XSLT).
- [F16H2] The Perl script **nuno** is a utility for creating sequences of unique numbers and is used to generate an XML containing local names used to construct URIs for unresolved people and organization URIs. The script is self-describing when invoked as '**nuno u**' on the bash shell command line.
- [F16H3] This directory is where RDF is placed when produced. There are files for new people and organizations and of course for the **vivo: EducationalTraining** instances (**ED.rdf**).
- [F16H4] The command sequence that produces the RDF is found here.
- [F16H5] This directory contains the result set as an XML in the file **EduRS.xml**.
- [F16H6] This directory contains the intermediate XML files produced along the way to the RDF.
- [F16H7] This directory contains the XSLTs used to transform **EduRS.xml** into **ED.rdf**. Notice that the file **saxon9he.jar** is the transform engine that executes the XSLTs.

[Start](#) [Previous](#) [Next](#)