XSLT Ingest Example: Create RDF

Create RDF

At this point we have everything needed to generate the RDF for each **EduRecord**. We will create RDF/XML that describes each as an instance of the class **vivo:EducationalTraining**. This object is identified by a URI as are the educational institution conferring the degree and the recipient. Each instance is further enriched by other data properties. The special **local:weakAttribution** predicate is used when the recipient has no netid. The full transform can be found in **mkEduRDF.xsl**, the main parts are shown in the next two figures.

```
<xsl:template match='/EduRecords'>
<rdf:RDF>
<xsl:value-of select='concat($NL,"   ")'/>

<xsl:for-each select='./EduRecord'>

<xsl:value-of select='concat($NL,"   ")'/>                    0
<xsl:comment>
<xsl:value-of select='concat("EduRec ",position())'/>
</xsl:comment>
<xsl:value-of select='concat($NL,"   ")'/>

<rdf:Description rdf:about="{edUri}" >
<rdf:type rdf:resource=
       'http://vivoweb.org/ontology/core#EducationalTraining'/>    1

<local:degreeLevel><xsl:value-of select='normalize-space(edDeg)'/>
</local:degreeLevel>                                          2

<xsl:if test='not(nid) or nid = ""'>                          3
<local:weakAttribution>
<xsl:value-of select='"true"'/>
</local:weakAttribution>
</xsl:if>

<rdfs:label>                                                  4
<xsl:value-of select='normalize-space(concat(edDeg,", ",edMajor))'/>
</rdfs:label>

<core:majorField><xsl:value-of select='edMajor'/></core:majorField>  5
```

mkEduRDF.xsl Fragment 1 - Figure 15a

- [F15H0] Pretty printing that produces nicely formatted output.
- [F15H1] Declare the type of each instance.
- [F15H2] Using a **local:degreeLevel** predicate, assert the degree level.
- [F15H3] Assert a weak attribution when the **EduRecord** has no netid.
- [F15H4] Assert a label for each instance.
- [F15H5] Declare the major field of study.

```
<xsl:call-template name='addDtiYear'>                                    6
<xsl:with-param name='uri' select='concat(edUri,"-DTV")'/>
<xsl:with-param name='year' select='edYear'/>
</xsl:call-template>

<core:trainingAtOrganization>                                            7
<xsl:comment><xsl:value-of select='edSchool'/></xsl:comment><xsl:text>
</xsl:text>
<xsl:value-of select='"          "'/>
<rdf:Description rdf:about="{edSchoolUri}"></rdf:Description>
</core:trainingAtOrganization>

<core:educationalTrainingOf>                                             8
<xsl:comment>
<xsl:value-of select='concat(ln,", ",fn," ",mn," (",nid,")")'/>
</xsl:comment><xsl:text>
</xsl:text>
<xsl:value-of select='"          "'/>

<rdf:Description rdf:about="{personUri}">
<core:educationalTraining rdf:resource="{edUri}" />
</rdf:Description>
</core:educationalTrainingOf>

</rdf:Description>                                                       9
</xsl:for-each>
</rdf:RDF>
<xsl:text>
</xsl:text>
</xsl:template>
```

mkEduRDF.xsl Fragment 2 - Figure 15b

- [F15H6] Call a named template (found in **auxfuncs.xsl**) to create a **vivo:DateTimeInterval** for the year the degree was conferred.
- [F15H7] Assert the organization RDF for each instance (with pretty-print code).
- [F15H8] Assert the degree recipient RDF (with pretty-print code).
- [F15H9] Close the loop and finish up.

The results of this step can be found in **example/rdf/ED.rdf** .

Create RDF for New Persons and Organizations

This part of the process is needed to provide RDF that describes the new people and organizations discovered in earlier steps. The transforms are **mkAccPeopleRdf.xsl** and **mkAccOrgRdf.xsl** . These transforms are fairly simple and are applied to the **NewPers.xml** and **NewOrgs.xml** respectively. The output of these transformations can be found in **example/rdf** in the files **NewPers.rdf** and **NewOrgs.rdf** . These RDF files must be asserted using the VIVO user interface so that the data model of Figure 2 can be traversed and so that when **Per0.xml** and **Org0.xml** are reconstructed for the next round of ingest the new people and organizations names together with their URIs are available for matching and assignment. Otherwise duplicates with different URIs are likely to be created.

Add Predicates and RDF to VIVO

The VIVO user interface should now be used to add the two new predicates to the local ontology before adding the RDF files. If this isn't done first you won't get another chance until the RDF in **ED.rdf** is retracted. At this point the **ED.rdf** file can be asserted, followed by the **NewPers.rdf** and **NewOrgs.rdf** files.

Start Previous Next