Installing CSV2RDF4LOD

----------------------------------

First, you must install csv2rdf4lod before you can convert your data, this is available at:

<https://github.com/timrdf/csv2rdf4lod-automation/wiki/Installing-csv2rdf4lod-automation>

### Please follow every instruction very carefully. For example, after you run the install.sh script, don’t forget to set your variable CSV2RDF4LOD\_BASE\_URI.

In my-csv2rdf4lod-source-me.sh, set variable CSV2RDF4LOD\_BASE\_URI to <http://sparql.tw.rpi.edu/> as follows:

export CSV2RDF4LOD\_BASE\_URI="http://http://sparql.tw.rpi.edu/"

### Make sure you “source csv2rdf4lod-automation/my-csv2rdf4lod-source-me.sh” before doing any conversion, or make sure you have this line included in your ~/.bashrc file!

Also as indicated, make sure the directory path you create is within a directory called source.

### If you don’t understand something please email [lebot@rpi.edu](mailto:lebot@rpi.edu) and CC me [seyeda2@rpi.edu](mailto:seyeda2@rpi.edu).

Says a bit about it, and describes some commands that help you orient:

<https://github.com/timrdf/csv2rdf4lod-automation/wiki/directory%20conventions>

SETTING UP YOUR PROJECT AREA

----------------------------------------------------

How to create a directory structure according to the "source-dataset-version" convention:

<https://github.com/timrdf/csv2rdf4lod-automation/wiki/Conversion-process-phase%3A-retrieve>

<https://github.com/timrdf/csv2rdf4lod-automation/wiki/A-quick-and-easy-conversion>

When the instructions tell you to run cr-create-convert-sh.sh, if your data file is delimited with something other than commas, you need to change your delimiter as an option. For example if you delimiter is “|” you run the command as follows:

cr-create-convert-sh.sh -w --delimiter \| source/101920122453t8PM.tx

for tab its \\t

CONFIGURING ENHANCEMENTS

------------------------------------------------

For an enhancement on the properties used, where we want the property to use a vocabulary already used within the project:

<https://github.com/timrdf/csv2rdf4lod-automation/wiki/conversion:equivalent_property>

So for example, we want to enhance the 5th column of our data file which has the header “Common Name”, we edit the file in directory manual/ that ends in e1.params.ttl, and edit the text:

conversion:enhance [

ov:csvCol 5;

ov:csvHeader "Common Name";

#conversion:label "Common Name";

conversion:comment "";

conversion:range todo:Literal;

];

to include the “conversion:equivalent\_property” and what you want the property to be, here its wildlife:hasCommonName:

conversion:enhance [

ov:csvCol 5;

ov:csvHeader "Common Name";

#conversion:label "Common Name";

conversion:comment "";

conversion:equivalent\_property wildlife:hasCommonName;

conversion:range todo:Literal;

];

If you are introducing a new namespace, here it is wildfile, you must edit that file and also the raw.ttl file to include that prefix line, as follows:

@prefix wildlife: <http://www.semanticweb.org/ontologies/2012/2/wildlife.owl#> .

After you make these changes you are ready to re-run the shell script and then look in the directory automatic/ for a file that ends in e1.ttl, which will be your rdf file with the enhancements applied.

For another enhancement, where we want to have a typing for the resource within the respective cell:

<https://github.com/timrdf/csv2rdf4lod-automation/wiki/conversion%3Adomain_name>

enhancement for not converting blank fields:

conversion:interpret [

conversion:symbol "";

conversion:interpretation conversion:null;

];

for more details see:

<https://github.com/timrdf/csv2rdf4lod-automation/wiki/conversion:interpret>

converting date to xsd:

<https://github.com/timrdf/csv2rdf4lod-automation/wiki/conversion:datetime_pattern>

combining columns to construct xsd:dateTime type:

conversion:enhance [

ov:csvCol 11;

ov:csvHeader "Date";

a conversion:Omitted;

#conversion:label "Date";

conversion:comment "";

conversion:range todo:Literal;

];

conversion:enhance [

ov:csvCol 12;

ov:csvHeader "Start Time";

#conversion:label "Start Time";

conversion:eg "colum 11= 20120101 | column 12= 01:00";

conversion:range\_template "[#11][#12]Z"; # Concatenate cols 11 and 12

conversion:pattern "yyyyMMddHH':'mmZ"; # Specify how to read the concatenation

conversion:range xsd:dateTime; # Cast the value to a dateTime

];

gives:

:thing\_3

e1:start\_time "2011-12-31T19:00:00-05:00"^^xsd:dateTime ;

for converting date of form 2012-01-10 use the enhancement conversion pattern:

conversion:enhance [

ov:csvCol 3;

ov:csvHeader "date";

#conversion:label "date";

conversion:comment "";

conversion:eg "2012-01-01";

conversion:pattern "yyyy-mm-dd"; #### <--- This is what I added. should be MM according to <https://github.com/timrdf/csv2rdf4lod-automation/wiki/conversion:datetime_pattern>

conversion:range xsd:date;

];

One row of air measurements:

<https://github.com/timrdf/csv2rdf4lod-automation/blob/master/doc/examples/source/lebot/structural-domain-name-subclass-of/version/2012-Oct-24/source/measurements.csv>

with the enhancements:

<https://github.com/timrdf/csv2rdf4lod-automation/commit/af68408f60af6606df0f2627bd6ded3e5af31ac7>

which includes the following two enhancements:

conversion:enhance[

conversion:domain\_template “am\_[r]’;

conversion:domain\_name “AirMeasurement”;

];

conversion:enhance[

conversion:class\_name “AirMeasurement”

conversion:subclass\_of “pol:AirMeasurement”;

];

produces the RDF:

@prefix pol: <<http://escience.rpi.edu/ontology/smenateco/2/0/pollution.owl#>> .

:am\_2 rdfs:label "am\_2" ;

dcterms:identifier "am\_2" ;

coin:slug "am\_2" ;

dcterms:isReferencedBy <<http://localhost/source/lebot/dataset/structural-domain-name-subclass-of/version/2012-Oct-24>> ;

void:inDataset <<http://localhost/source/lebot/dataset/structural-domain-name-subclass-of/version/2012-Oct-24>> ;

a pol:AirMeasurement , local\_vocab:AirMeasurement ;

# ^^^ notice pol:AirMeasurement here, the class that you want these rows to be typed to.

An enhancement of typing what is in the range to class geospecies:FamilyConcept is as follows:

conversion:enhance [

ov:csvCol 13;

ov:csvHeader "Family";

#conversion:label "Family";

conversion:comment "";

conversion:equivalent\_property geospecies:inFamily;

conversion:range\_name "FamilyConcept";

conversion:range rdfs:Resource;

];

conversion:enhance [

conversion:class\_name "FamilyConcept";

conversion:subclass\_of geospecies:FamilyConcept;

];

result:

e1:date "2180-08-10"^^xsd:date ;

e1:location typed\_site:Troy ;

e1:value "14.6"^^xsd:decimal ;

ov:csvRow "2"^^xsd:integer .

typed\_site:Troy dcterms:identifier "Troy" ;

a local\_vocab:Site , pol:Site ;

rdfs:label "Troy" .

Tim, do you have any idea if the format '13-FEB-09' can be typed as xsd:date ?

conversion:pattern "dd-MMM-yy";

conversion:range xsd:date;

Another enhancement:

If I wanted to replace a cell with a class from an external

ontology, what would that enhancement look like?

conversion**:**enhance **[**  
 ov**:**csvCol 1**;**  
 conversion**:**interpret **[**  
 conversion**:**symbol "Senate State"**;**  
 conversion**:**interpretation **<**http**://**dbpedia**.**org/resource/United\_States\_Senate**>;**  
 **];**

2) This one is two-part, how do I assert existential restriction on this?

measurement1 a (measuresChemical some CarbonMonoxide)

so say my raw conversion gives me "measurement1

pollution:measuresChemical "Carbon monoxide" " and I want the

assertion given above.

Define a class in your pol: ontology “CarbonMonoxideMeasurement equivalentClass (measuresChemical some nci:CarbonMonoxide)” and then

then:

conversion**:**enhance **[**  
 ov**:**csvCol 1**;**  
 conversion:equivalent\_property rdfs:type  
 **<**http**://**dbpedia**.**org/resource/United\_States\_Senate**>;**

conversion**:**interpret **[**  
 conversion**:**symbol "Carbon Monoxide"**;**  
 conversion**:**interpretation **pol:**CarbonMonoxideMeasurement **];**

[https://github.com/timrdf/csv2rdf4lod-automation/wiki/conversion%3Aobject\_search](https://github.com/timrdf/csv2rdf4lod-automation/wiki/conversion:object_search)

allows you to search the cell value, match substrings with a regex, and construct triples from the captured groups in the regex.

This is done IN ADDITION TO the standard processing for the cell value.

for converting US-RI-002 to 002 for the literal:

my enhancement:

conversion:enhance [

ov:csvCol 1;

ov:csvHeader "identifier";

#conversion:label "identifier";

conversion:eg "US-VA-019";

conversion:comment "";

conversion:range todo:Literal;

conversion:object\_search [

conversion:regex "^(..)-(..)-(.\*)$";

conversion:predicate foaf:homepage;

conversion:object "[/s]id/[\\3]";

];

];

my output:

:thing\_2

e1:identifier "US-VA-019" ; #### <--- This comes from the "standard processing"

foaf:homepage <<http://localhost/source/lebot/id/019>> ; #<---- This is added by the object\_search.

use equiv property and interpret for a column

for setting the uri of the dataset, use the following enhancement, which is simply adding the default namespace prefix for the conversion file. (base\_uri is where you are going to deploy it in e.g. logd.tw.rpi.edu)

conversion:base\_uri “<http://sparql.tw.rpi.edu>”^^xsd:anyURI;

(this is within the dataset container but before the conversion\_process container)

for provenance linking the dataset to the webpage it was taken from:

<http://sparql.tw.rpi.edu/source/epa-air/dataset/carbon-monoxide/version/2012-Oct-23> foaf:homepage <http://www.epa.gov/ttn/airs/airsaqs/detaildata/downloadaqsdata.htm> .

Fields that Qi is working on:

**e1:order "Falconiformes" ;**

**e1:family "Accipitridae" ;**

**e1:genus "Accipiter" ;**

**e1:specific\_epithet "cooperii" ;**

**wildlife:has\_Higher\_Geography "United StatesFloridaPinellas" ;**

**wildlife:has\_Country "United States" ;**

**wildlife:has\_State\_Province "Florida" ;**

**wildlife:has\_County "Pinellas" ;**

**wildlife:has\_Locality "33702 Saint Petersburg" ;**

**e1:decimal\_latitude "27.8449898" ;**

**e1:decimal\_longitude "-82.6429214" ;**

**wildlife:has\_Year\_Collected "2009" ;**

**wildlife:has\_Month\_Collected "2" ;**

**wildlife:has\_Day\_Collected "13" ;**

**wildlife:has\_Time\_Collected "11" ;**

**e1:julian\_day "44" ;**

**wildlife:has\_Collector "GBBC Participant 2" ;**

**wildlife:has\_Project\_Code "GBBC" ;**

**wildlife:has\_Protocol\_Type "GBBC" ;**

**wildlife:has\_Protocol\_Code "P26" ;**

**e1:protocol\_species\_targeted "Birds" ;**

**e1:survey\_area\_identifier "L131280" ;**

**e1:sampling\_event\_identifier "S4562058" ;**

**e1:duration\_in\_hours "3.5" ;**

**wildlife:has\_Number\_Observers "2" ;**

**wildlife:has\_Observation\_Count "1" ;**

**wildlife:has\_Obs\_Count\_At\_Least "1" ;**

**wildlife:has\_Obs\_Count\_At\_Most "1" ;**

**e1:observation\_date "13-FEB-09" ;**

**e1:all\_individuals\_reported "yes" ;**

**e1:all\_species\_reported "yes" ;**

**e1:common\_name "Cooper's Hawk" ;**

**e1:record\_permisions "5 public" ;**

**e1:taxonomic\_authority\_authors "eBird" ;**

**e1:taxonomic\_authority\_version "1.051" ;**

**e1:taxonomic\_authority\_year "2011" ;**

**e1:record\_review\_status "Valid but not reviewed" ;**

**ov:csvRow "3"^^xsd:integer .**

Fields that Chris does not have a typing for:

AQS Site ID

POC

CBSA

Local Site Name

Address

Datum

Lat/Lon Accuracy (meters)

Elevation (meters MSL)

Monitor Start Date

Last Sample Date

Monitor Still Active

Measurement Scale

Measurement Scale Definition

Sample Duration

Sample Collection Frequency

Sample Collection Method

Sample Analysis Method

Method Reference ID

FRM/FEM?

Monitor Objective

Monitor Type

Reporting Agency

Download Annual Data

Download Daily Data