Using the Convert CSV to RDF ingest tool

This guide describes just one (rather primitive) way to ingest data into VIVO. See https://wiki.duraspace.org/x/MYUQAg for a detailed discussion of data ingest.

Introduction

This guide will walk through the use of the 'Convert CSV to RDF' tool, a semi-automated method of converting comma separated or tab separated text files into RDF that can be displayed in VIVO. These files should include one row of data per record (e.g., a person or publication) and represent the fields or properties associated with each record in separate columns within the row, much as the values appear in a spreadsheet. The most common pattern of loading CSV files involves one CSV file per type of data to be loaded. Note, the current ingest tools involve working through a number of steps from original source data files to the appearance of new data in VIVO. The process requires some understanding of semantic web data modeling and some training.

Access the tool by navigating to the Site Administration section, clicking Ingest tools, then Convert CSV to RDF.

Mapping Ontologies to other Ontologies

When VIVO's ingest tools read in a CSV file, the data read from the file will be stored in the VIVO database as an extra "model," or secondary database managed by the Jena semantic web libraries underlying VIVO.

The next step is to link imported data sets using information stored with the source data. If an object not previously in VIVO has been found, a new record (individual) must be created in VIVO using the CONSTRUCT form of query in the SPARQL query language for RDF.

The query looks at the list of imported data and inserts statements creating a new individual wherever no match to an existing individual is found. The query of imported data is expressed using the ontology of the import; the CONSTRUCT statements use the class and property names of VIVO ontology. This accomplishes the mapping between the source ontology and the VIVO ontology.

When populating VIVO for the first time, all the data from a dataset can be added from the imported Jena model to VIVO by a CONSTRUCT query that again translates from the RDF in the imported model to the RDF in VIVO. When a dataset has already been mapped to data and there's a likelihood that the new data being imported may already exist in VIVO, the process becomes more complicated due to the need to match each prospective import against existing data.

Example workflow

The first time through an ingest process is a slow process of evaluating the source data, deciding what cleanup will be needed, and working through each step. The following example details each step in the process and how the VIVO software supports that step. As the process expands to include requirements to match against existing data, additional steps must be introduced to test for matches and perform different actions, usually as successive steps identified through different queries.

For this example, a sample set of data for people, their positions, and the organizations at which their positions reside has been provided at http://sourceforge.net/projects/vivo/files/Data%20Ingest/. The guide gives instructions on how to ingest the data, map the data to the VIVO/ISF ontology, and load the data into an RDF format. Upon completion linked data in VIVO will include people to positions and the positions to organizations.

Process:

1. Prepare CSV file
2. Create workspace models for ingesting and constructing data
3. Pull CSV file into RDF
4. Confirm data property URIs and RDF structure
5. Convert temporary RDF into VIVO/ISF ontology using SPARQL CONSTRUCT
6. Load data to the current web model

Step 1: Prepare CSV file

CSV template files can be downloaded here (or from the Source Forge links included):

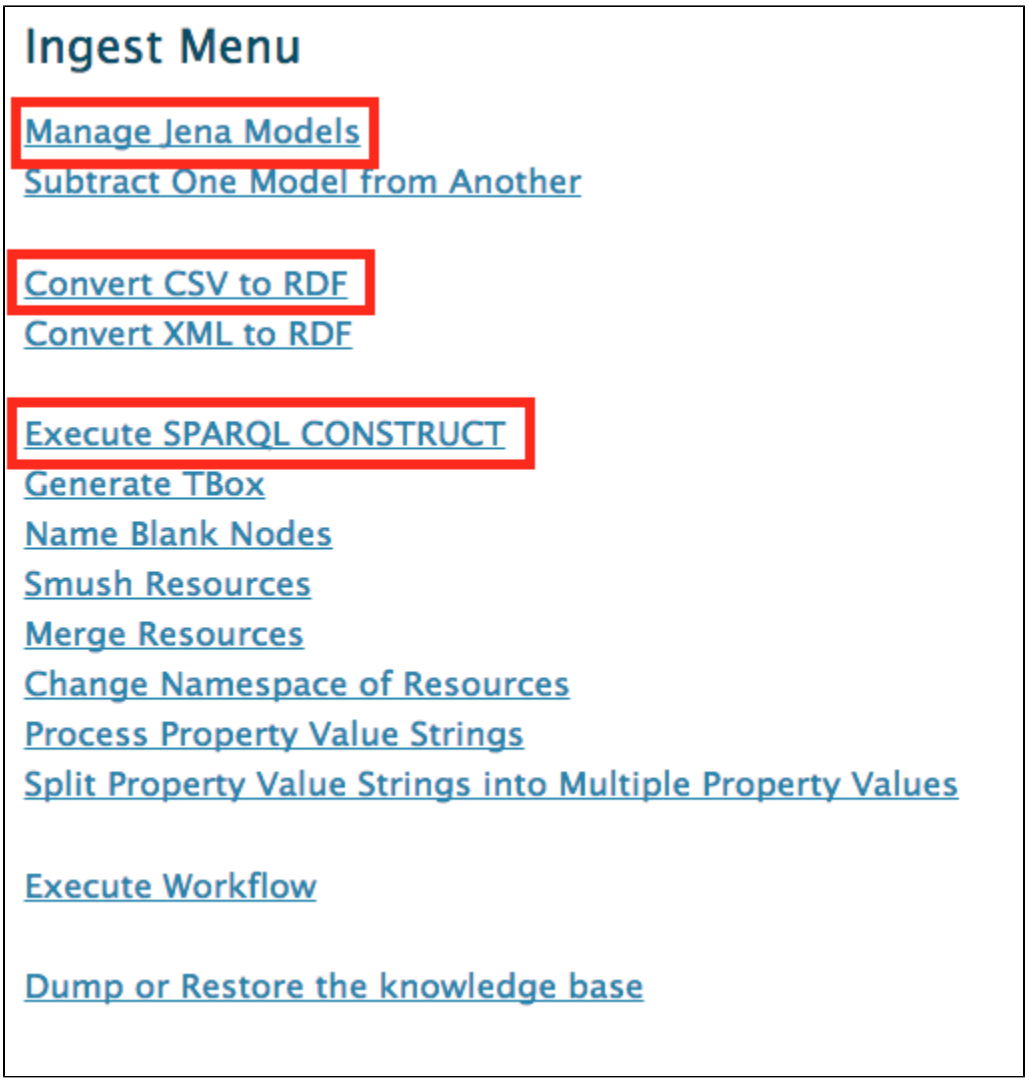
- organization.csv (or http://sourceforge.net/projects/vivo/files/Data%20Ingest/organization.csv/download)
- position.csv (or http://sourceforge.net/projects/vivo/files/Data%20Ingest/position.csv/download)
- people.csv (or http://sourceforge.net/projects/vivo/files/Data%20Ingest/people.csv/download)

For the purposes of this walkthrough, you can leave the csv files as is if you wish.

Note: You can optionally create a local ontology and class specifically for the 'person_ID' and 'org_ID' fields included in the example CSV files. See the 'Add New Data Properties' section of the Ontology Editors Guide here: https://wiki.duraspace.org/x/2AQGAg.

Step 2: Create Workspace Models

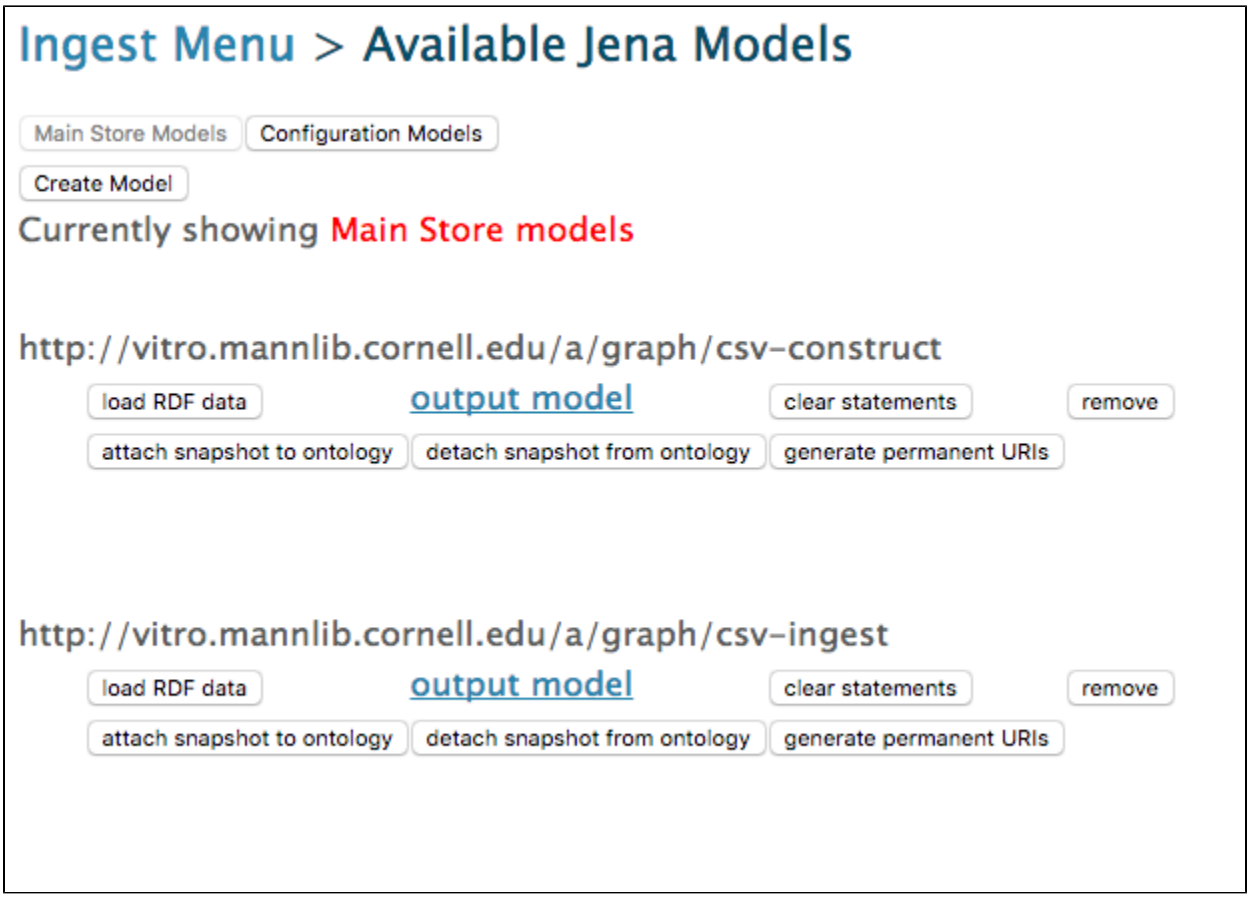
Highlighted in red are the three Ingest Menu options we will be using for this demonstration.



We will create two temporary data models titled 'csv-ingest' and 'csv-construct' to keep our work separate from the main VIVO models.

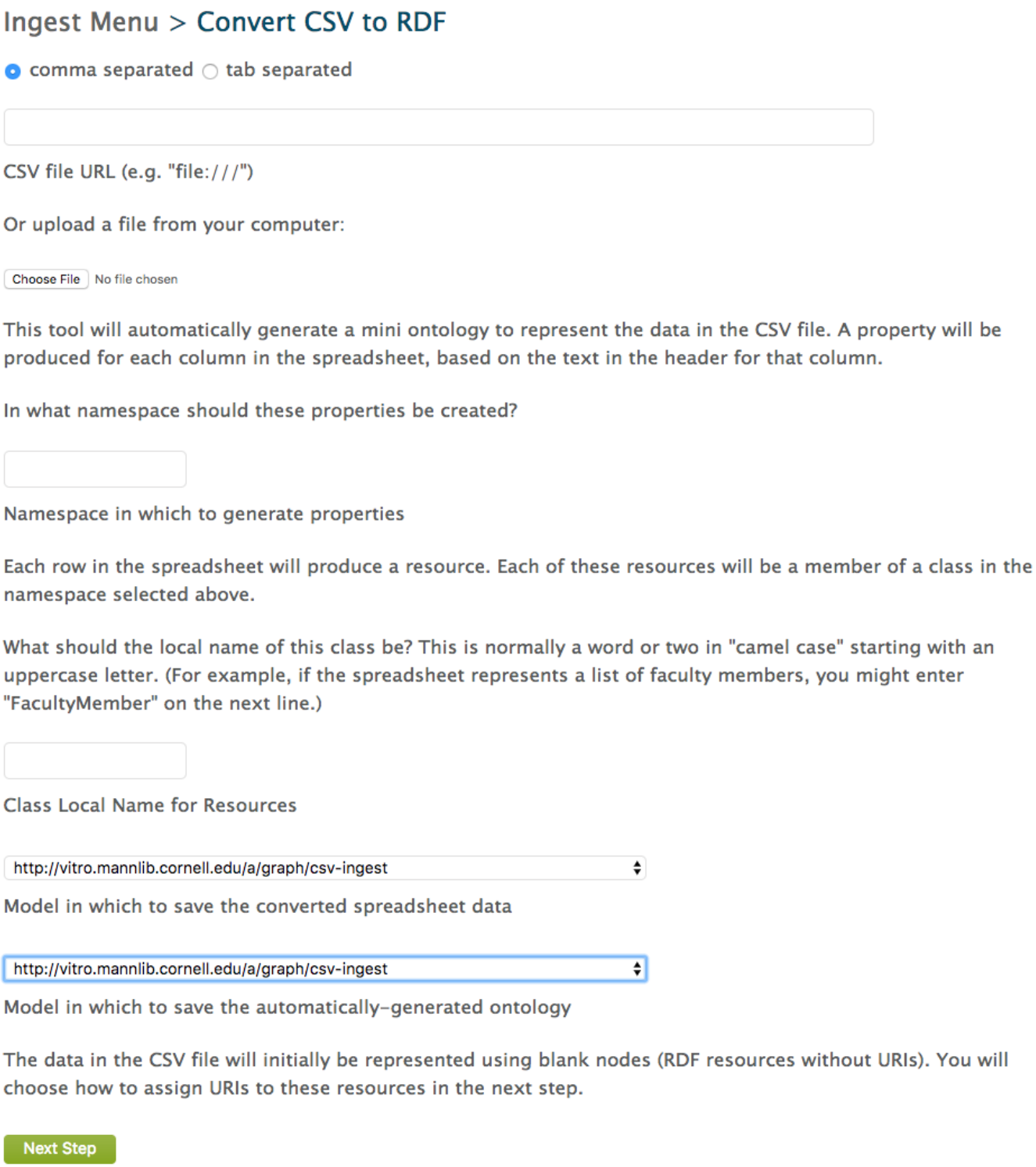
1. Select "Ingest Tools" from the Advanced Tools Menu
2. Select "Manage Jena Models"
3. Click the "Create Model" button then type in a name for your model, 'csv-ingest'.
4. Repeat step 3 and name the model 'csv-construct'

You should now see your new models on the main 'Manage Jena Models' page.



Step 3: Pull CSV File into RDF

Now click 'Convert CSV to RDF' on the Ingest Menu. Begin by supplying a URL for your CSV file, or by uploading the file directly from your computer. Start with people.csv. Complete the fields in the form (explanation follows graphic below).



Namespace for Classes and Properties ('in what namespace should these properties be created?'

This namespace can be temporary since we will later map the tool's output to the VIVO/ISF ontology. For example, you can use

```
http://localhost/vivo/
```

Class Name

The class name is also a temporary value for this example. This value does not follow the created entity since you will shift the properties from the format they come in into the ontologies format. For this example, the suggested class names are “ws_ppl”, “ws_org”, and “ws_post”.

Destination Models

The data and ontology model option dropdown menus should list the model to ingest into. This is where we select one of those 'workspace' models we created earlier, "csv-ingest" for example.

After clicking convert we can check our RDF conversion through two methods, the first method being the quickest.

1. From the Ingest Menu, select the 'Manage Jena Models' and then select the ingest model's 'output model.' This is something you can open with WordPad and see the created triples for your CSV file.
2. Also from the Manage Jena Model we can attach the ingest model to the current webapp. Then we can go back to the Site Admin, select Class Hierarchy link, select 'All Classes' and navigate to the Class Name you created (individual, for example).

Click 'Next Step.' Here you will create the URI for your new individuals.

Select URI prefix

It is most convenient if this matches the URL for your VIVO instance plus '/individual/', e.g. http://vivo.university.edu/individual/.

URI suffix

This can be a random number or a created based on a pattern plus the value from your CSV file.

Now, click 'Convert CSV.' The CSV data should now be converted into temporary RDF and inserted into the 'csv-ingest' model.

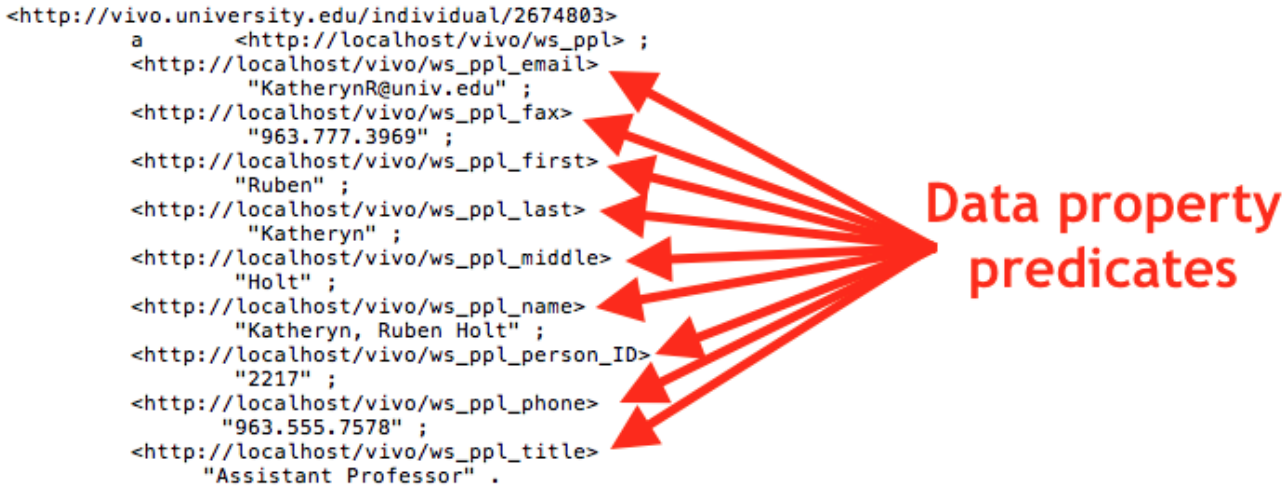
You can now repeat step 3 for both organizations.csv and positions.csv before continuing to assure positions will be attached to both people and organizations, or for simplicity, you can ignore organizations and positions for now.

Step 4: Confirm data property URIs and RDF structure

The CSV to RDF tool converts the CSV into temporary RDF that we can query using SPARQL. This temporary RDF cannot be displayed in VIVO. We will transform the RDF into the VIVO/ISF ontology format in the next step. First, take a look at the temporary RDF we have created.

Ingested Data URIs

Confirm the data property URI's that were created for each of the columns in your csv file by navigating back to the 'Manage Jena Models' page and clicking 'output model' below csv-ingest. A description of the format is described in the figure below. The predicates are the properties we need for our SPARQL Query. If you used the 'http://localhost/vivo/' format used above, it should look similar to this:



We will use these data property predicates (e.g. <http://localhost/vivo/ws_ppl_email>) in the 'WHERE' part of the next step.

Step 5: Convert temporary RDF into VIVO/ISF ontology using SPARQL CONSTRUCT

The 'Execute SPARQL CONSTRUCT' tool described in this step appears to not work correctly in the latest versions of VIVO. You can get around this by following the instructions below using the 'SPARQL Query' tool in the admin panel. Save the resulting RDF to a file (noting the format, any will work). Then, in lieu of Step 6, upload to VIVO using the Add/Remove RDF tool.

Now, we must query and CONSTRUCT new RDF that is mapped to the VIVO/ISF ontology. Diagrams to help visualize the VIVO/ISF ontology model are available on the wiki at https://wiki.duraspace.org/x/ycCdB. Some helpful links to information on SPARQL queries can be found at https://wiki.duraspace.org/x/lwUGAg. Basically, we will query the database for the triples in Step 4 to pull out the resource URIs and store them in variables (e.g. http://vivo.university.edu/individual/2674803 ?person) in the WHERE part of the query. We then CONSTRUCT new triples that are in the proper VIVO/ISF format using those variables for the resource URIs (e.g. ?person) and data values (e.g. ?fullname).

Return to the ingest menu and click 'Execute SPARQL CONSTRUCT'
Example SPARQL query for people.csv:

```
CONSTRUCT {

?person <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://vivoweb.org/ontology/core#FacultyMember> .
?person <http://www.w3.org/2000/01/rdf-schema#label> ?fullname .

?person <http://purl.obolibrary.org/obo/ARG_2000028> ?vcard .
?vcard <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2006/vcard/ns#Individual> .
?vcard <http://www.w3.org/2006/vcard/ns#hasName> ?vcard_name .
?vcard_name <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2006/vcard/ns#Name> .

?vcard_name <http://www.w3.org/2006/vcard/ns#givenName> ?first .
?vcard_name <http://vivoweb.org/ontology/core#middleName> ?middle .
?vcard_name <http://www.w3.org/2006/vcard/ns#familyName> ?last .

?vcard <http://www.w3.org/2006/vcard/ns#hasEmail> ?vcard_email .
?vcard_email <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2006/vcard/ns#Email> .
?vcard_email <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2006/vcard/ns#Work> .
?vcard_email <http://www.w3.org/2006/vcard/ns#email> ?email .
}

WHERE {

?person <http://localhost/vivo/ws_ppl_name> ?fullname .

OPTIONAL { ?person <http://localhost/vivo/ws_ppl_first> ?first . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_middle> ?middle . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_last> ?last . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_email> ?email . }
?person <http://localhost/vivo/ws_ppl_person_ID> ?hrid .
BIND(URI(CONCAT(STR( ?person ), "_vcard")) AS ?vcard ) .
BIND(URI(CONCAT(STR( ?person ), "_vcardname")) AS ?vcard_name ) .
BIND(URI(CONCAT(STR( ?person ), "_vcardemail")) AS ?vcard_email ) .

}
```

Next:

- Select Source Model (‘csv-ingest’)
- Select Destination Model (‘csv-construct’)
- Select "Execute CONSTRUCT"
- Upon completion, the system will report ‘n statements CONSTRUCTed’.

Note, the BIND statements in the query allow us to create unique URIs for the associated vCard objects required in VIVO v1.6+.

Step 6: Load to Webapp

The live webapp that is indexed does not allow for models to just be attached. Attaching models works well for seeing what we have constructed or ingested or smushed, but those models are lost when Tomcat refreshes.

The final step is to output the final model and add it into the current model

1. From the Ingest Menu, select "Manage Jena Models"
2. Click "output model" below 'csv-construct'
3. Save the resulting file
4. Navigate back to the Site Administration page
5. Select Add/Remove RDF
6. Browse to the file previously saved
7. Select N3 as import format*
8. Confirmation should state ‘Added RDF from file people_rdf. Added 415 statements.’

- The output engine uses N3, not RDF/XML. This is important to note when adding the 'output mode' RDF data into the webapp. RDF/XML is the default setting for the drop down list as most ontologies are written in RDF/XML.

The ingested data should now display in both the index and the search results. It is part of the main webapp and will be retained upon a Tomcat restart. The founding steps of data ingest have been completed. Repeat steps 4 and 5 for organizations and people using the SPARQL queries supplied in the appendix below.

Appendix A: SPARQL Queries

People Construct:

```
CONSTRUCT {

?person <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://vivoweb.org/ontology/core#FacultyMember> .
?person <http://www.w3.org/2000/01/rdf-schema#label> ?fullname .

?person <http://purl.obolibrary.org/obo/ARG_2000028> ?vcard .
?vcard <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2006/vcard/ns#Individual> .
?vcard <http://www.w3.org/2006/vcard/ns#hasName> ?vcard_name .
?vcard_name <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2006/vcard/ns#Name> .

?vcard_name <http://www.w3.org/2006/vcard/ns#givenName> ?first .
?vcard_name <http://vivoweb.org/ontology/core#middleName> ?middle .
?vcard_name <http://www.w3.org/2006/vcard/ns#familyName> ?last .

?vcard <http://www.w3.org/2006/vcard/ns#hasEmail> ?vcard_email .
?vcard_email <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2006/vcard/ns#Email> .
?vcard_email <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.w3.org/2006/vcard/ns#Work> .
?vcard_email <http://www.w3.org/2006/vcard/ns#email> ?email .
}

WHERE {

?person <http://localhost/vivo/ws_ppl_name> ?fullname .

OPTIONAL { ?person <http://localhost/vivo/ws_ppl_first> ?first . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_middle> ?middle . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_last> ?last . }
OPTIONAL { ?person <http://localhost/vivo/ws_ppl_email> ?email . }
?person <http://localhost/vivo/ws_ppl_person_ID> ?hrid .
BIND(URI(CONCAT(STR( ?person ), "_vcard")) AS ?vcard ) .
BIND(URI(CONCAT(STR( ?person ), "_vcardname")) AS ?vcard_name ) .
BIND(URI(CONCAT(STR( ?person ), "_vcardemail")) AS ?vcard_email ) .

}
```

Organization Construct:

```
CONSTRUCT {

?org <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> ?type_uri .
?org <http://localhost/vivo/ontology/vivo-local#orgID> ?deptID .
?org <http://www.w3.org/2000/01/rdf-schema#label> ?name .
}


WHERE {
?org <http://localhost/vivo/ws_org_org_ID> ?deptID .
?org <http://localhost/vivo/ws_org_org_name> ?name .
?org <http://localhost/vivo/ws_org_org_vivo_uri> ?type .

BIND(URI(?type) as ?type_uri)
}
```

Basic Position to People & Organization Construct:

Note: The example query does not account for start dates

```
CONSTRUCT {

?position <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://vivoweb.org/ontology/core#FacultyPosition> .
?position <http://www.w3.org/2000/01/rdf-schema#label> ?poslabel .
?position <http://vivoweb.org/ontology/core#relates> ?org .
?position <http://vivoweb.org/ontology/core#relates> ?person .

}
WHERE {

?org <http://localhost/vivo/ws_org_org_ID> ?deptID .
?org <http://localhost/vivo/ws_org_org_name> ?name .
?position <http://localhost/vivo/ws_post_department_ID> ?postOrgID .
?org <http://localhost/vivo/ws_org_org_ID> ?orgID .
FILTER((?postOrgID)=(?orgID))


?position <http://localhost/vivo/ws_post_department_ID> ?orgID .
?position <http://localhost/vivo/ws_post_person_ID> ?posthrid .
?position <http://localhost/vivo/ws_post_job_title> ?poslabel .
?person <http://localhost/vivo/ws_ppl_person_ID> ?perhrid .
FILTER((?posthrid)=(?perhrid))

}
```