Maintaining RDF Vocabularies in Spreadsheets

Gregg Kellogg
gregg@greggkellogg.net
@gkellogg
http://ruby-rdf.github.io/presentations/Maintaining-RDF-Vocabularies-DC2017/index.html

Origins of Practice

- Spreadsheets have been used (an abused) for many purposes over time.
  - Convenient way to edit and maintain small amounts of tabular data
  - … Sometimes grows to too much data
  - … Sometimes used for non-tabular data

Origins of Practice

- W3C maintains many small vocabularies
- Specifications may define namespaces with terms

- Test Suites may extend other test suites and/or need to reduce to available datasets for processing (e.g., EARL reports)

Uses of Vocabularies

- Machine-readable metadata
  - Classes, Properties, Datatypes, Domains, Ranges
- Human-readable
  - Developer Documentation
- Generate/maintain JSON-LD Contexts
  - Context can include the full vocabulary definition extractable as RDF

General Approaches

- No vocabulary document (described in prose only)
- Manage RDF serializations by hand (RDF/XML, Turtle, JSON-LD)
- Use Protégé
- Create alternate representations automatically
    - Possible for RDF/XML, Turtle, N-Triples
    - Awkward for RDFa or JSON-LD
- Use a convenient non-RDF representation to derive other formats
    - Spreadsheets!

No vocabulary

- Example: RDFa Processor Status*

- Advantages: Simple for editor

- Disadvantages: No machine-readable representation

- An `rdfa:Error` must be generated when the document fails to be fully processed as a result of non-conformant Host Language markup.
- A `rdfa:Warning` must be generated when a CURIE prefix fails to be resolved.
- A `rdfa:Warning` must be generated when a Term fails to be resolved.

* https://www.w3.org/TR/rdfa-core/#processor-status

Hand-built vocabulary

- Example: RDF Schema*
  - Advantages: Simpler for spec editor.
  - Disadvantages: HTML and N-Triples not maintained together. HTML not machine-readable

All things described by RDF are called *resources*, and are instances of the class `rdfs:Resource`. This is the class of everything. All other classes are subclasses of this class. `rdfs:Resource` is an instance of `rdfs:Class`.

```
rdfs:Resource a rdfs:Class ;
  rdfs:isDefinedBy <http://www.w3.org/2000/01/rdf-schema#> ;
  rdfs:label "Resource" ;
  rdfs:comment "The class resource, everything." .
```

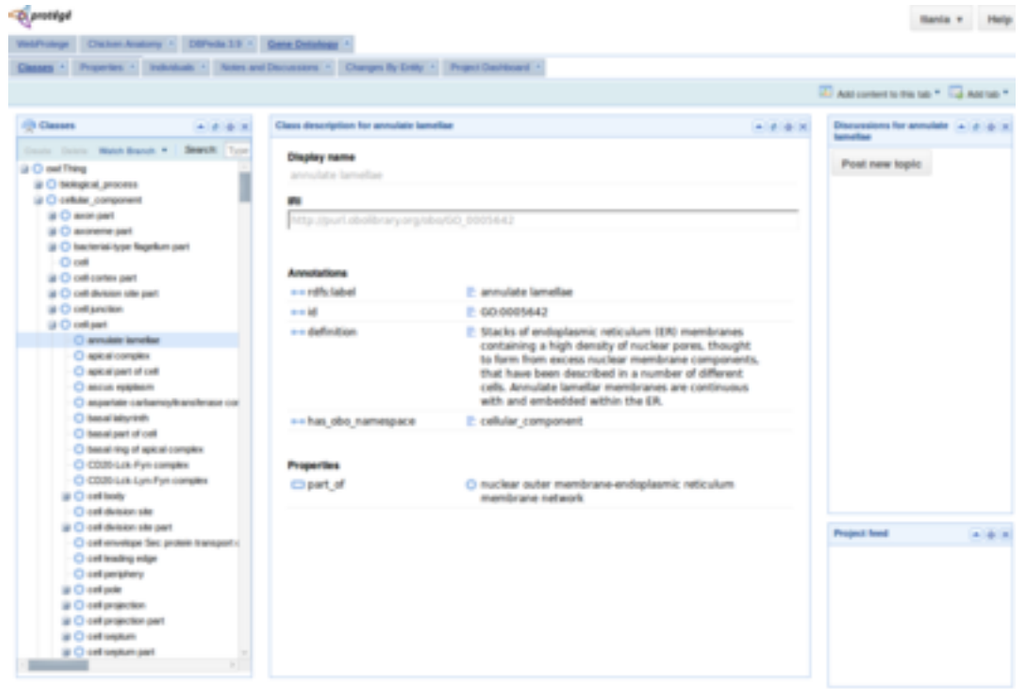
* https://www.w3.org/TR/2014/REC-rdf-schema-20140225/#ch_resource

Protégé

- Great tool for managing complex vocabularies

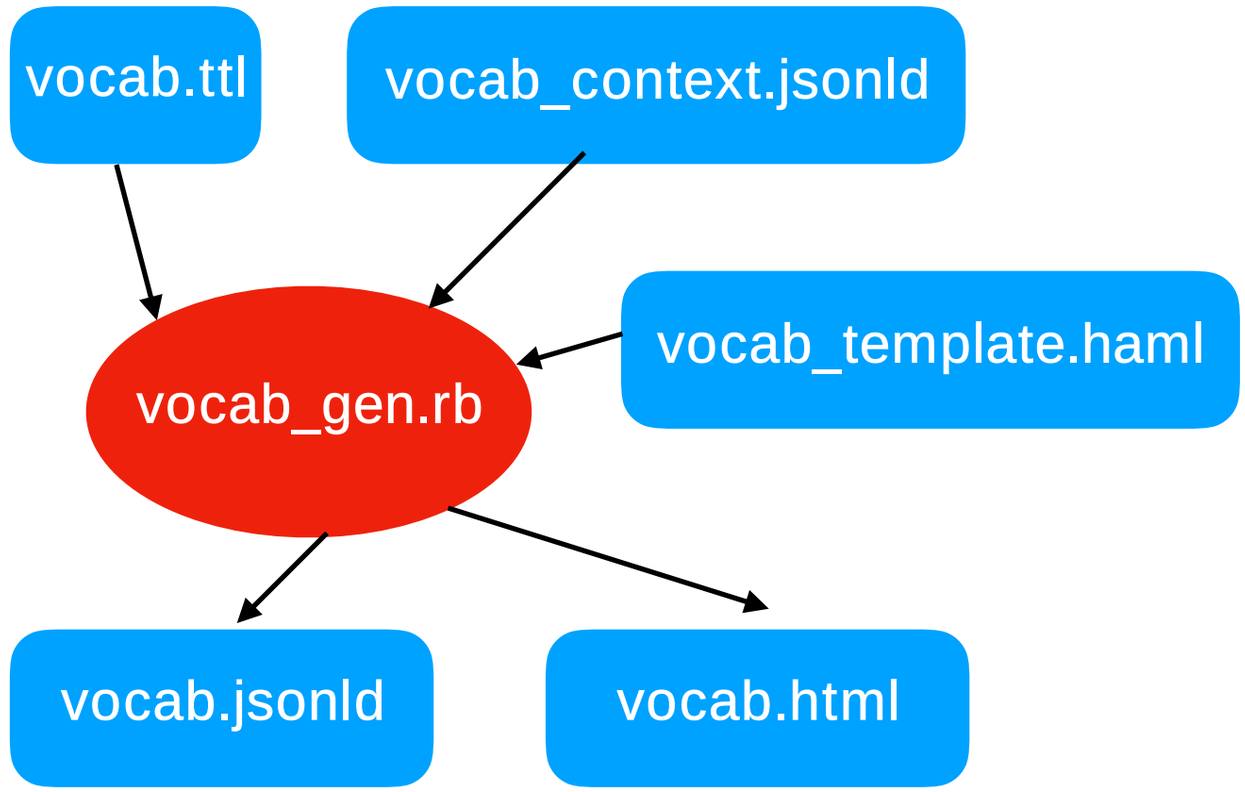
- Advantages: WYSIWYG ontology development. Fully supported tool.

- Disadvantages: Little control over serialization. Details can get lost in UI.



RDF Source Document

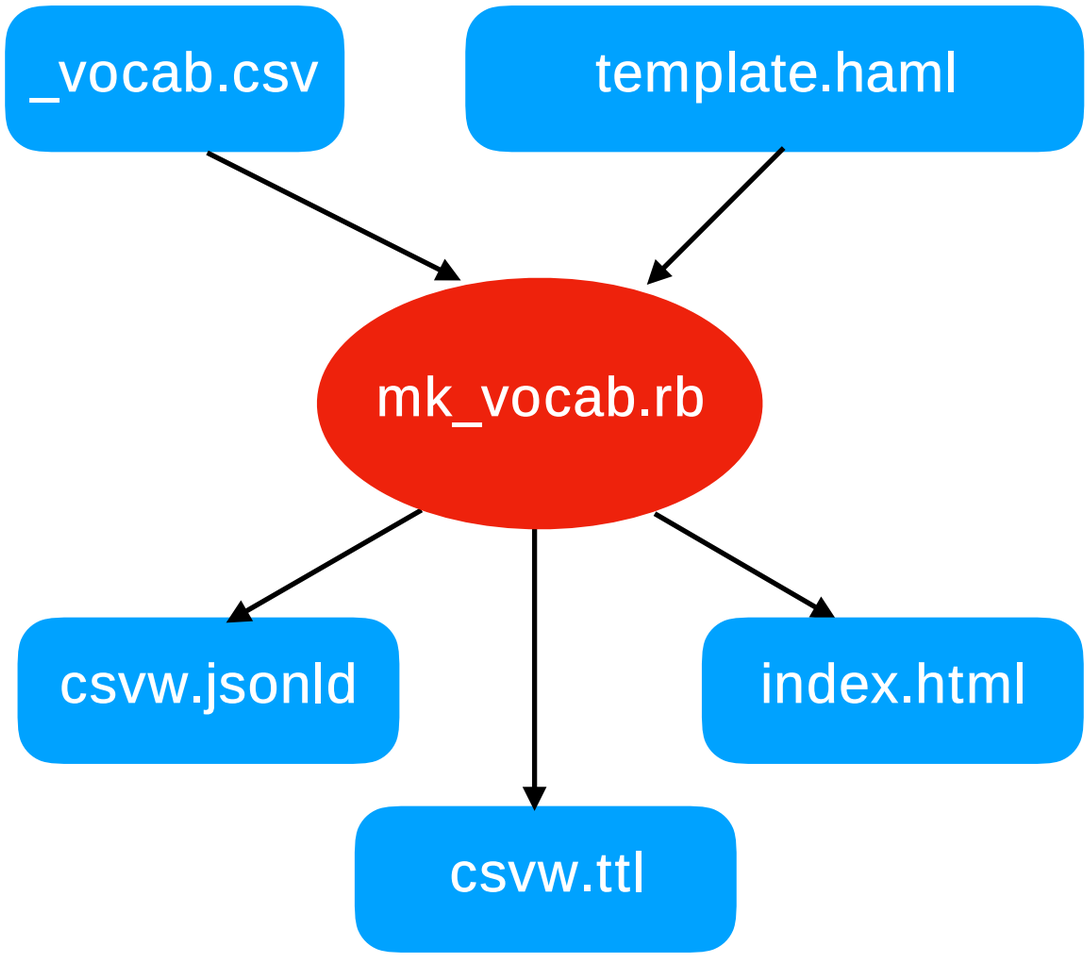
- Example: JSON-LD Test-Suite*
  - Hand maintain Turtle RDFS document and JSON-LD context
  - Direct transform Turtle to JSON-LD
  - Use Haml template to generate HTML from JSON-LD.
  - Advantages: Single source, consistency
  - Disadvantages: Context made by hand. JSON-LD may not be convenient.



* https://github.com/json-ld/json-ld.org/tree/master/test-suite

Vocabulary in CSV

- Examples: CSVW[1], ShEx[2]
  - Information in spreadsheet
  - script creates JSON object indexed by type
  - after processing, objects gathered into JSON-LD to describe both context and vocabulary
  - Custom Turtle generation
  - Haml template-based HTML generation



[1] https://github.com/w3c/csvw/tree/gh-pages/ns
[2] https://github.com/shexSpec/shexspec.github.io/tree/master/ns

RDF.rb serialization support

- Ruby RDF.rb internalizes RDFS+ vocabularies for convenience and reasoning
- Read and RDF serialization to create Vocabulary
- Vocabulary supports .to_ttl, .to_jsonld, and .to_html with reasonable defaults which may be customized