RDF Validation @ DPLA

Use Cases & Approaches
Harvest/Enrichment Pipeline

Abstracting the Pipeline

➢ Harvester
➢ Mapping
➢ Enrichment

Harvest
Unprocessed XML, RDF, JSON, etc...
Map
DPLA MAP Record

Validate
Enrich
Metadata Application Profile
# Metadata Application Profile

## DPLA MAP classes and properties

A. \texttt{class= "dpla:SourceResource"}

This is a subclass of \texttt{"edm:ProvidedCHO,"} which comprises the SRs (in EDM called \texttt{"cultural heritage objects"} or \texttt{"CHOs")} about which DPLA collects descriptions. These descriptions are of the SRs, not the digital representations of them.

<table>
<thead>
<tr>
<th>Label</th>
<th>Property, JSON-LD expression</th>
<th>Sub-property of</th>
<th>Range</th>
<th>Usage</th>
<th>Value Type, Origination</th>
<th>Vocab Schema</th>
<th>Syntax Schema</th>
<th>Obligation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection*</td>
<td>dct:isPartOf, .sourceResource.collection</td>
<td></td>
<td>dc:relation</td>
<td>dcmitype: Collection</td>
<td>URI of collection or aggregation of which SR is a part</td>
<td>URI, DPLA</td>
<td></td>
<td>min 0, max unbounded</td>
</tr>
<tr>
<td>Contributor</td>
<td>dc:contributor, .sourceResource.contributor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Creator*</td>
<td>dc:creator, .sourceResource.creator</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date*</td>
<td>dc:date, .sourceResource.date</td>
<td></td>
<td>edm: TimeSpan</td>
<td>Date value as supplied by data provider</td>
<td>Literal and/or URI, Partner</td>
<td>W3CDTF, EDTF, local</td>
<td>min 0, max unbounded</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>dc:description, .sourceResource.description</td>
<td></td>
<td></td>
<td>Includes but is not limited to: an abstract, a table of contents, or a free-text account of SR</td>
<td>Literal, Partner</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extent</td>
<td>dct:extent, .sourceResource.extent</td>
<td></td>
<td></td>
<td>Size or duration of SR</td>
<td>Literal, Partner</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Format*</td>
<td>dc:format,</td>
<td></td>
<td></td>
<td>File format, physical medium or dimensions</td>
<td>Literal and/or</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Are These Constraints?

➢ Required & Recommended Fields, Maximum Cardinality (“Obligation”)
➢ “Range” (Value Class/Type Restrictions)
➢ Vocabulary Schema
➢ Syntax Schema
➢ Origination
DPLA Use Cases

Validating Enrichments

➢ Harvested/mapped data not expected to conform

➢ DPLA enrichments add (domain model) structure, external links, syntax and datatype transformations, etc...

➢ Validate compliance against those constraints

Does DPLA’s data conform to its model?
DPLA Use Cases

Levels of Validation

➢ Communicate expectations & errors to data providers

➢ Drive quality control metrics & reporting

Does partner supplied data meet harvest and quality requirements?
Linked Data & API Consumers

➤ Developers should be able to confirm that data conforms to expectations
➤ Validate as JSON or JSON-LD via API
➤ Validate as RDF

*Can consumers interact with data in both Open & Closed worlds.*
Validation Scope

Scope of AP/Validations

Domain “Controlled Vocabs”

Geographic Data

Annotations

etc...
Graph ⇔ AP ⇔ Object Model

- (RDFS) Class ⇔ (OO) Class
- Resource ⇔ Object
- Predicate ⇔ Property
- RDF Literal Datatypes ⇔ Language Constructs
Graph $\Rightarrow$ AP $\Rightarrow$ Object Model
Graph ⇔ AP ⇔ Object Model
Simple Validations

class SourceResource < ActiveTriples::Resource
  configure :base_uri => 'http://dp.la/api/items/',
              :type => 'http://dp.la/about/map/SourceResource'

  validates_presence_of :rights, :title
  validates_type_of :date,
                    :type => 'http://www.europeana.eu/schemas/edm/TimeSpan'

  property :title, :predicate => RDF::DC11.title
  property :rights, :predicate => RDF::DC11.rights
  property :date, :predicate => RDF::DC11.date,
               :class_name => 'DPLA::TimeSpan'

  ...
end
Validating Class

➢ “Domain”
  ➢ Object validates based on its declared class
  ➢ Do we need predicate-based validation?

➢ “Range”
  ➢ validates_rdf_type_of

What about Sub/Superclasses? Inference is possible, but unwieldy.
Validating Vocabularies

validates_xyz_of

➢ vocabulary
➢ vocabulary_scheme
➢ namespace