Applied Linked Geo. Metadata

Samvera Connect 2019

James Griffin
Princeton University Library

Getting Started

- GitHub (SSH Keys)
- git
- Ruby (RVM, unless you are using something else)
- Code Editors (Atom, VSCode, Sublime, vim, emacs, ed...)
- Browsers
  - Please use Chrome/Brave or Firefox
- Network Connection
  - VPNs should be fine, but please do not use Tor
- Slack

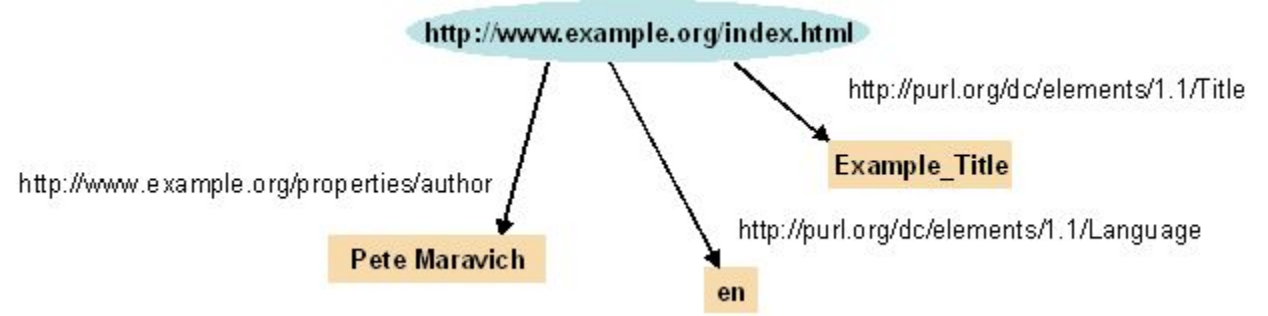
Overview of Samvera Applications

- Introduction to/Review of the Samvera Application Stack
- Fedora
- Solr
- Ruby on Rails
- ActiveFedora (or Valkyrie, but we will discuss that later)
- Hyrax/Avalon/CurationConcerns/Sufia/Custom App.

Linked Data

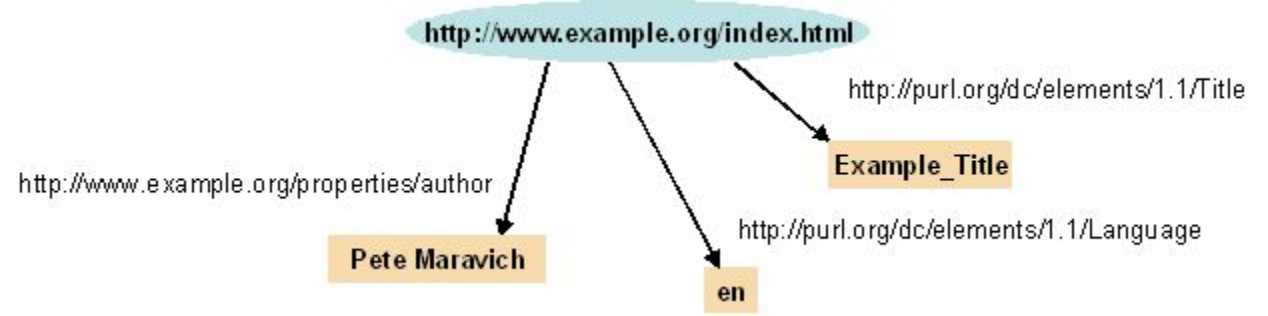
- What is linked data?
- Definitions vary, but we will focus on the Resource Description Framework
- RDF uses graphs (as in networks): Unidirectional and acyclic
- RDF Graphs are composed of **3-tuples (triples)**

```
<https://institution.edu/blog/1> <http://purl.org/dc/elements/1.1/Title> “Our Samvera Blog Post”
```



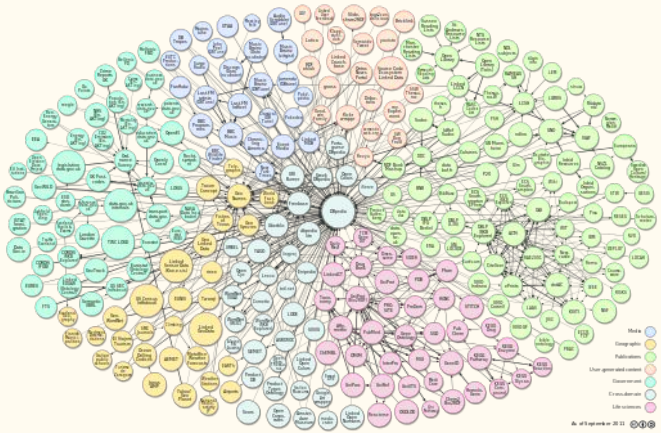
Linked Data

- **Triples** are logical assertions about entities with URL-like identifiers
  - These are generalized into URIs (Uniform Resource Identifiers) or IRIs (Internationalization)
- Syntax: Subject, Predicate, Object
- Triples are expressed using the Resource Description Framework (RDF)



Linked Data

- Why linked data?
- If we all use the same URIs for the same entities, we can all build off of shared data models
- **Global data interoperability**
- Published to the World Wide Web, linked data provides a semantic layer to browsing the WWW
- The WWW (ideally) becomes a global database
- **This becomes the semantic web**



Querying Linked Data

- Linked Data is stored in triple or quad stores (generalized as “graph stores”)
- In order to extract information, one uses the SPARQL query language
- SPARQL Protocol and RDF Query Language is a standard maintained by the World Wide Web consortium
- One uses it to extract information from graphs:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?name
WHERE {
    ?person foaf:name ?name .
    ?person foaf:knows ?friend .
}
```

Querying Linked Data

- RDF itself supports XML data structures
  - Strings
  - Integers
  - Date/Time stamps
- **It cannot be used to encode spatial information**

Querying Spatial Linked Data

- Support for encoding spatial information has been gradual
- **GeoSPARQL extends SPARQL for spatial querying**
- GeoSPARQL provides predicates and spatial data structures for triples

```
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>

SELECT ?myMapUrl
WHERE {
    ?myMapUrl dc11:coverage ?fGeom .
        ?fGeom geo:sfWithin '''
        <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
            Polygon ((-83.4 34.0, -83.1 34.0,
                      -83.1 34.2, -83.4 34.2,
                      -83.4 34.0))
        '''^^geo:wktLiteral
}
```

- Because GeoSPARQL is new, it requires graph stored which support it

GraphDB

- For this workshop, GraphDB will be hosted in the cloud
- **It consumes GBs of memory, please do not run it locally**
- It is currently deployed using a Docker image:
  - https://github.com/jrgriffiniii/graphdb-docker
  - https://hub.docker.com/r/jrgriffiniii/graphdb
- *If you want to run this locally, this *must* have CORS disabled*

What about Samvera?

- GraphDB and GeoSPARQL are not unique to Samvera
- Samvera repositories (Hyrax) use linked data for persistence
  - This is how metadata is stored and retrieved
- This is largely due to Fedora

Fedora

- Fedora, properly-speaking, is a repository itself
- It does not have a rich interface
  - (Demo here)
- Fedora builds off of the Linked Data Platform standard
- Basically, Fedora accomplishes the following using linked data:
  - Stores files
  - Stores metadata for the files
  - Performs fixity checks for the files
  - Captures “audit trails”
- Hyrax uses Fedora for storage, but then builds user-facing features (dashboards, user permission management, object viewers, derivative generation...)

Fedora

- Fedora is **not** a graph store (you cannot query it for data with SPARQL)
- Fedora **can** synchronize with a graph store
- For this workshop, we synchronize with GraphDB
- This work is also on GitHub:
  - https://github.com/jrgriffiniii/fcrepo-graphdb-docker
  - https://github.com/jrgriffiniii/linked-geo-metadata-docker
- **Let's take a look at Fedora triples in GraphDB**

Hyrax

- Hyrax is a Samvera repository solution bundle
- It is aimed towards serving as an institutional repository (theses and dissertations) or cultural heritage object repository (digitized manuscripts)
- Developed to handle a selection of media types
  - High resolution images (TIFFs, JPEGs, PNGs)
  - Documents (PDFs, Word Documents)
  - Audiovisual material (MPEG4)
- Provide support for self-deposit workflows
  - Users can review the submissions of others
- Provide extensible content modeling

Hyrax

- As mentioned, Fedora is used to store metadata and files for Hyrax
- What we store in Hyrax is accessible from Fedora
- What we store in Hyrax can be synchronized by Fedora with GraphDB
- **(This is accomplished using the fcrepo-camel-toolbox project)**

Hyrax and GraphDB

- The objective of the first set of these tasks in the workshop is to synchronize Hyrax Works with GraphDB
- *This will involve developing custom a custom content model with Hyrax*

Getting Started with Hyrax

- Starting Hyrax
- *Please find a custom Hyrax implementation at:* https://github.com/jrgriffiniii/vesconte
- We need to first use a supported version of Ruby (2.6.2)
- We need to connect to the Fedora and Solr servers in AWS
  - James will provide the IP addresses and ports
- Please start the Rails servers
- Please visit http://localhost:3000

Getting Started with Hyrax

- One should now see the landing page 🎉
- This is a Hyrax implementation branded *Vesconte*
- Hyrax itself is a framework
  - *We can revisit how Hyrax implementations are built later*
- A few more steps are needed to proceed
  - **Creating an Admin. Set**
  - **Signing Up**
  - **Granting yourself administrative privileges**

Pietro Vesconte, Genoese Cartographer from 14th century

Getting Started with Vesconte

Depositing a Scanned Map

- Please select “Deposit My Work”
- Please download a downsampled image from Google Drive
- (We will be using a Princeton map item as an example)
- *James will guide you through the metadata and ingestion form*

Getting Started with Vesconte

Depositing a Second Scanned Map

- Thank you for depositing a Scanned Map
- We are now going to proceed with a second example
- This is another Princeton map item
- Please download the image for this item also using Google Drive

Linked Data from Vesconte

- Hyrax has ingested these objects along with our metadata
- How do we access the linked data for these works?

GraphDB

- Please visit `http://[HOST_ADDRESS]:7200`
- You will now be faced with the GraphDB Workbench
- *James will guide through accessing the linked data for the works*

Linked Data from Vesconte

- Now that you have seen the linked data in GraphDB
- **Let's evaluate how this was enabled in Vesconte**

Adding a Work Type to Vesconte

- A practical example might suffice here
- *(Please prepare your code editors)*
- Let us generate a *RasterDataset* Work Type!

```
bundle exec rails generate hyrax:work RasterDataset
```

- This generates the necessary files in order to support a new content model
- **You will need to restart the Rails server here**

Adding a RasterDataset Type to Vesconte

- Let us add metadata fields for this new work type
- (Open `app/models/raster_dataset`)
- Add a new property for `coverage`:

```
property :coverage, predicate: ::RDF::Vocab::DC11.coverage, class_name: 'WellKnownTextLiteral'
```

- What is WellKnowTextLiteral?
- *James will explain how this works with Hyrax and Fedora*

Adding a Bounding Box to RasterDataset

- This was ported from GeoWorks
- The field is overridden in `app/views/records/edit_fields/_coverage.html.erb`
- *James will explain how this was integrated*

Depositing a RasterDataset

- From the landing page, deposit a new work
- For this example, we are going to use a Stanford dataset
- The raster for this item can be downloaded from Google Drive
- After depositing this data set, please return to the GraphDB workbench
- `http://[HOST_ADDRESS]:7200`

Querying for Spatial Data

- Now that we have our linked data indexed, let's try GeoSPARQL queries
- Please visit the query interface at `http://[HOST_ADDRESS]:7200/sparql`
- *We can craft a query for retrieving our data set*

Querying for Spatial Data

```
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX geof: <http://www.opengis.net/def/function/geosparql/>
PREFIX fcrepo: <info:fedora/fedora-system:def/model#>
PREFIX dc11: <http://purl.org/dc/elements/1.1/>
PREFIX dc: <http://purl.org/dc/terms/>
PREFIX hydra: <http://projecthydra.org/works/models#>

SELECT *
WHERE {
  ?subject a hydra:Work ;
     fcrepo:hasModel ?model ;
     dc:title ?title ;
     dc11:coverage ?coverage .
  OPTIONAL {
    ?subject  dc11:description  ?abstract
  }
  FILTER (geof:sfWithin(?coverage, '''
    <http://www.opengis.net/def/crs/OGC/1.3/CRS84>
      [...]
    '''^^geo:wktLiteral))
}
```

Let's explore the components of this

- PREFIX
- SELECT and WHERE ?subject
- dc11:coverage
- OPTIONAL
- FILTER
- geof:sfWithin
- geo:wktLiteral

Querying for Spatial Data

- Well-Known Text
  - Geospatial format for specifying points, lines, paths, and polygons
  - Varies in implementation, but generally standardized
  - Open Geospatial Consortium provided this in a specification
- CRS
  - Coordinate Reference System
  - How grid lines are mapped to a model of the globe
  - CRS84 references a standard model (geodetic datum) used commonly by GIS applications
  - RDF references a URI in order to link to this model

Querying for Spatial Data

- Well-Known Text
  - Geospatial format for specifying points, lines, paths, and polygons
  - Varies in implementation, but generally standardized
  - Open Geospatial Consortium provided this in a specification
- CRS
  - Coordinate Reference System
  - How grid lines are mapped to a model of the globe
  - CRS84 references a standard model (geodetic datum) used commonly by GIS applications
  - RDF references a URI in order to link to this model

Querying for Spatial Data

- Spatial functions
  - GeoSPARQL offers a set of predicates for querying spatial relationships
  - `geof:sfWithin` functions are just for a polygon or point within another spatial object
  - There are other functions for spatial relationships:
    - `geof:sfContains`
    - `geof:sfDisjoint`
    - `geof:sfEquals`
    - `geof:sfIntersects`
- Spatial predicates
  - There are also corresponding predicate forms used for making assertions (rather than for filtering through result sets: e. g. `geo:sfIntersects`)
- *Please reference the GeoSPARQL specification for more detail*

Consuming Spatial Data

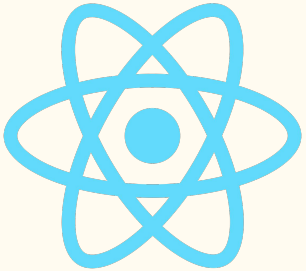
- A brief overview has been given of publishing spatial linked data
- This overview has attempted to also minimally address querying the data
  - These can be explored in greater depth
- **However, providing an example for consuming this data might be valuable**

Building a Linked Data Portal

Glitch

- Glitch is a hosting service for JavaScript applications
- Please visit https://glitch.com/edit/#!/mature-peony in your web browser
- *James will guide you in authenticating on Glitch*
- This Glitch project is a small React/Redux portal for linked data

Building a Linked Data Portal



- React
  - JavaScript framework developed by Facebook
  - Provides a component-driven approach to developing user interfaces
  - Lightweight and flexible, meant to be integrated with other JavaScript frameworks

Building a Linked Data Portal

- Redux
  - Open source JavaScript framework
  - Frequently paired with React
  - Addresses application layers below the user interface
  - *For our purposes, one uses Redux to query the GraphDB*

Building a Linked Data Portal

- Architecture of the Data Portal
  - Leaflet
  - React
  - Redux
  - GraphDB

Building a Linked Data Portal

- Leaflet
  - Geospatial and map visualization library (GeoBlacklight uses this)
  - Using Redux, we...
    - Retrieved Works from the GraphDB installation
    - Transform the spatial metadata elements into GeoJSON
    - Add the GeoJSON objects to the map

Building a Linked Data Portal

- Leaflet and React
  - React wraps the Leaflet map
- Leaflet, React, and Redux
  - React “listens” for application events using Redux
  - Redux emits actions and routes them to update the application state
  - Example: User moves the map, and a new search for the region is executed

Building a Linked Data Portal

- Redux and GraphDB
  - Redux is also responsible for querying the GraphDB installation
  - Within a Redux action is where on crafts the GeoSPARQL query
  - *James will guide through the GeoSPARQL integration for Redux*

Extending a Linked Data Portal

- This is a minimal linked data portal
  - Glitch permits you to “remix” a project
    - (Copy this project and add your own modifications)
    - *Please, experiment with this!*
  - Faceting, text search, authentication and user management are all possible
  - **Let’s retrieve some more metadata by editing our Glitch**

Reviewing the Stack

- Questions
  - About Fedora?
  - About GraphDB?
  - About Hyrax?
  - About React/Redux?
  - About other layers?

Geo Linked Data

Thank you for your time, attention, and patience!

Attributions

1. Richard CyganiakAnja Jentzsch, *LOD Cloud Diagram As of September 2011.*
2. Vladimir Agafonkin, CloudMade, *Leaflet logo.*
    a. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:
    1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
    2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.