LINKED ARCHIVAL METADATA

A GUIDEBOOK

Technically speaking, it is about URIs and ontologies. Practically speaking, it is simply a new way for cultural heritage institutions to do the same thing they have always been doing.

Eric Lease Morgan

February 6, 2014

draft Draft Draft! DRAFT! D R A F T ! ! !

Executive Summary

[The Executive Summary will list core objectives, anticipated outcomes, and implications that will provide administrators or other senior leaders with the information that they will need in order to understand the benefits and potential costs of this path.]

Introduction

Linked Archival Metadata: A Guidebook provides archivists with an overview of the current linked data landscape, define basic concepts, identify practical strategies for adoption, and emphasize the tangible payoffs for archives implementing linked data. It focuses on clarifying why archives and archival users can benefit from linked data and will identify a graduated approach to applying linked data methods to archival description.

The Guidebook is a product of the Linked Archival Metadata planning project (LiAM), led by the Digital Collections and Archives at Tufts University and funded by the Institute of Museum and Library Services (IMLS). LiAM's goals include defining use cases for linked data in archives and providing a roadmap to describe options for archivists intending to share their description using linked data techniques.

Why linked data, and why now?

Linked data, or more recently referred to as "linked open data" for reasons to be explained later, is a proposed technique for generating new knowledge. It is intended to be a synergy between people and sets of agreed upon computer systems that when combined will enable both people and computers to discover and build relationships between seemingly disparate data and information to create and discover new knowledge.

In a nutshell, this is how it works. People possess data and information. They encode that data and information in any number of formats easily readable by computers. They then make the encoded data and information available on the Web. Computers are then employed to systematically harvested the encoded data. Since the data is easily readable, the computers store the data locally and look for similarly encoded things in other locally stored data sets. When similar items are identified relationships can be inferred between the items as well as the other items in the data set. To people, some of these relationships may seem obvious and "old hat". On the other hand, since the data sets can be massive, relationships that were never observed previously may come to light, thus new knowledge is created.

Some of this knowledge may be trivial. For example, there might be a data set of places -- places from all over the world including things like geographic coordinates, histories of the places, images, etc. There might be another data set of poeple. Each person may be described using their name, their place of birth, and a short biography. These data sets may contain ten's of thousands of items each. Using linked data it would be possible to cross reference the people with the places to discover who might have met whom when and where. Some people may have similar ideas, and those ideas may have been generated in a particular place. Linked data may help in discovering who was in the same place at the same time and the researcher may be better able to figure out how a particular idea came to fruition.

Here's an example hitting closer to the home of archives and archivists. Suppose most archival finding aids were written in a format easily readable by computers. Let's call this format Encoded Archival Description. Let's suppose these finding aids were made available on the Web. Let's suppose one or more computers crawled these archival sites harvesting the finding aids. Once done a computer program could be used to find all the occurrences of particular name and generate a virtual finding aid that is more complete and more comprehensible than any single finding aid on that particular person.

The amount of data and information accessible today is greater in size than it has ever been in human history. Using our traditional techniques of reading, re-reading, writing, discussing, etc. is more than possible to learn new things about the state of the world, the universe, and the human condition. By exploiting the current state of computer technology is possible to expand upon our traditional techniques and possibly accelerate the mass of knowledge.

How to use the Guidebook

The structure of the Guidebook supports readers moving through the text in a variety of ways. Like a travel book, it provides useful high-level information for users who only need the basics, as well as in-depth information for those planning an extended stay in

LOD-land. The Guidebook is intentionally named, and will draw from the genre of actual travel guides (Fodors, etc.) providing readers easy access to both high-level information (know before you go, what to see if you're only there for a day) as well as in-depth details of for those staying in one place longer.

Synopses of the use cases developed by the LiAM project will be interspersed throughout the Guidebook to illustrate and frame the text. Each use case will be briefly described in 100-200 words with links to the full use cases on the LiAM website.

An initial release of the Guidebook will be in the form of a PDF document to be delivered to IMLS in fulfillment of the LiAM planning grant requirements as well as being shared with the public. However, the Guidebook's ongoing vitality will benefit from a more dynamic publication environment, and we therefore plan to publish it in a wiki connected to a code repository. This combination will enable updating of the resource to reflect changes in the field as well as providing a mechanism for sharing tools, scripts, and other code related to the project.

Much of the rest of the Guidebook, while providing a concise overview of today's linked data landscape and needs, would require ongoing updates, maintenance, and enhancement to describe implementation of LOD in the archival community over time.

Benefits

Linked data makes the content of archival collections more accessible and open doors for new types of service.
Archives are about collecting, organizing, preserving, and disseminating original, unique, and primary literature. These are the whats of archival practice, but the hows of archival practice evolve with the changing technology. With the advent of ubiquitous networked computing, people's expectations regarding access to information and knowledge have changed significantly. Unless institutions like archives change with the times, then the needs previously filled by archives will be filled by other institutions. Linked data is a how of archival practice, and it is one of those changes behooving archives to adopt. It is a standards-based technique for making data and information available on the Web. It is rooted in the very fabric of the Web and therefore is not

beholden to any particular constituency. It is a long lasting standard and practice that will last as long as the hypertext transfer protocol is operational.

Making archival descriptions and collection available via linked data will increase the use of those descriptions and collections. It is a form of benign advertising. Commercial search engines will harvest the linked data content and make it available it their search engines. Search engines will return hits to your descriptions and collections driving traffic to you and your site. Digital humanists will harvest your content, perform analysis against it, and create new knowledge or bring hidden knowledge to light. Computer scientist will collect your data, amalgamate it with the data of others, and discover relationship previously unconceived.
You can divide your combined collections and services into two tangible parts: 1) the collections themselves, and 2) the metadata describing them. It is usually possible to digitize your collections, but the result is rarely 100% satisfactory. Digitization is almost always a useful surrogate not a complete replacement. In this way, your collections as physical objects will always be a draw to all types of learners and researchers. The metadata, on the other hand, is 100% digitizable, and therefore lends itself very well to dissemination on the Internet. Linked data represents one way to make this happen. Few archival collections are 100% complete. There are always pieces missing, and some of those missing pieced will be owned by others. Your collections will have relationship with other collection, but you will not have direct access to those other collections. Some of these relationships are explicit. Some of them are implicit. If everybody were to expose their metadata then those explicit and implicit relationships can become more apparent. Once these relationships are strengthened and become more obvious, interest in the collections will increase accordingly, and the collections will be used to a greater degree. With this increased use will come increased attention, and in turn, a greater measure of success for the collections and services it provides.

Linked Data for Archives: a Primer

Linked Data is a process for manifesting the ideas behind Semantic Web. The Semantic Web is about encoding data, information, and knowledge in computer-readable fashions, making these encodings accessible on the World Wide Web, allowing computers to crawl the encodings, and finally, employing reasoning engines against them for the purpose of discovering and creating new knowledge. The canonical article describing this concept was written by Tim Berners-Lee, James Hendler, and Ora Lassila in 2001.

In 2006 Berners-Lee more concretely described how to make the Semantic Web a reality in a text called "Linked Data -- Design Issues". In it he outlined four often-quoted expectations for implementing the Semantic Web. Each of these expectations are listed below along with some of my own elaborations:

1. "Use URIs as names for things" - URIs (Universal Resource Identifiers) are unique identifiers, and they are expected to have the same shape as URLs (Universal Resource Locators). These identifiers are expected to represent things such as people, places, institutions, concepts, books, etc. URIs are monikers or handles for real world or imaginary objects.

2. "Use HTTP URIs so that people can look up those names." - The URIs are expected to look and ideally function on the World Wide Web through the Hypertext Transfer Protocol (HTTP), meaning the URI's point to things on Web servers.

3. "When someone looks up a URI, provide useful information, using the standards (RDF*, SPARQL)" - When URIs are sent to Web servers by Web browsers (or "user-agents" in HTTP parlance), the response from the server should be in a conventional, computer readable format. This format is usually a "serialization" of RDF (Resource Description Framework) -- a notation looking much like a rudimentary sentence composed of a subject, predicate, and object.

4. “Include links to other URIs. So that they can discover more things.” - Simply put, try very hard to use URIs that other people have have used. This way the relationships you create can literally be linked to the relationships other people have created. These links may represent new knowledge.

In the same text (“Linked Data -- Design Issues”) Berners-Lee also outlined a sort of reward system -- sets of stars -- for levels of implementation. Unfortunately, nobody seems to have taken up the stars very seriously. A person gets:

- 1 star for making data available on the web (in whatever format) but with an open license, to be Open Data

- 2 stars for making the data machine-readable structured data (e.g. excel instead of image scan of a table)

- 3 stars for making the data available in non-proprietary format (e.g. CSV instead of excel)

- 4 stars for using open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff

- 5 stars for linking your data to other people’s data to provide context

The whole idea works like this. Suppose I assert the following statement:

The Declaration Of Independence was authored by Thomas Jefferson.

This statement can be divided into three parts. The first part is a subject (Declaration Of Independence). The second part is a predicate (was authored by). The third part is an object (Thomas Jefferson). In the language of the Semantic Web and Linked Data, these combined parts are called a triple, and they are expected to denote a fact. Triples are the heart of RDF.

Suppose further that the subject and object of the triple are identified using URIs (as in Expectations #1 and #2, above). This would turn our assertion into something like this with carriage returns added for readability:

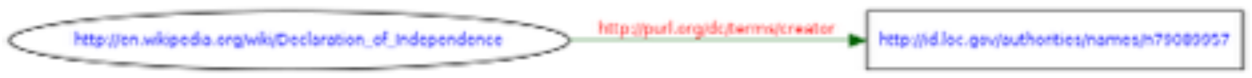
http://en.wikipedia.org/wiki/Declaration_of_Independence
was authored by
http://www.worldcat.org/identities/lccn-n79-89957

Unfortunately, this assertion is not easily read by a computer. Believe it or not, something like the XML below is much more amenable, and if it were the sort of content returned by a Web server to a Web browser (read “user-agent”), then it would satisfy Expectations #3 and #4 because the notation is standardized and because it points to other people’s content:

```
<?xml version=”1.0”?>
<rdf:RDF
  xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
  xmlns:dcterms=”http://purl.org/dc/terms/” >

  <!-- the Declaration Of Independence was authored by Thomas Jefferson -->
  <rdf:Description
  rdf:about=”http://en.wikipedia.org/wiki/Declaration_of_Independence”>
    <dcterms:creator>http://id.loc.gov/authorities/names/n79089957</dcterms:creator>
  </rdf:Description>

</rdf:RDF>
```


http://en.wikipedia.org/wiki/Declaration_of_Independence
http://purl.org/dc/terms/creator
http://id.loc.gov/authorities/names/n79089957

Suppose we had a second assertion:

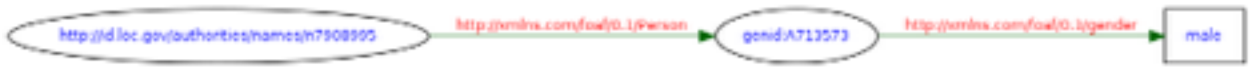
Thomas Jefferson was a man.

In this case, the subject is “Thomas Jefferson”. The predicate is “was”. The object is “man”. This assertion can be expressed in a more computer-readable fashion like this:

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">

  <!-- Thomas Jefferson is man (a male) -->
  <rdf:Description rdf:about="http://id.loc.gov/authorities/names/n7908995">
    <foaf:Person foaf:gender="male" />
  </rdf:Description>

</rdf:RDF>
```



http://id.loc.gov/authorities/names/n7908995
http://xmlns.com/foaf/0.1/Person
genid:A713573
http://xmlns.com/foaf/0.1/gender
male

Suppose there were smart Linked Data robot / spider. Suppose it crawled both Assertion #1 and Assertion #2, it then ought to be able to assert the following:

```
<?xml version="1.0"?>
<rdf:RDF
 xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/">

  <!-- the Declaration Of Independence was written by
  Thomas Jefferson, and Thomas Jefferson is a male -->
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/
Declaration_of_Independence">
 <dcterms:creator>
 <foaf:Person rdf:about="http://id.loc.gov/authorities/names/n79089957">
 <foaf:gender>male</foaf:gender>
 </foaf:Person>
 </dcterms:creator>
  </rdf:Description>

</rdf:RDF>
```

Looking at the two assertions, a reasonable person can deduce a third assertion, namely, the Declaration Of Independence was authored by a man. Which brings us back to the point of the Semantic Web and Linked Data. If everybody uses URIs (read "URLs") to describe things, if everybody denotes relationships (through the use of predicates) between URIs, if everybody makes their data available on the Web in standardized formats, and if everybody uses similar URIs, then new knowledge can be deduced from the original relationships.

Unfortunately and to-date too little Linked Data has been made available and/or too few people have earned too few stars to really make the Semantic Web a reality. The purpose of this guidebook is to provide means for archivists to do their part, make their content available on the Semantic Web through Linked Data, all in the hopes of facilitating the discovery of new knowledge. On our mark. Get set. Go!

There are a number of challenges in the process. Some of them are listed below, and some of them have been alluded to above:

Create useful LOD, meaning, create LOD that links to other LOD. LOD does not live in a world by itself. Remember, the "L" stands for "linked". For example, try to include URIs that are the URIs used on other LOD data sets. Sometimes this is not possible, for example, le with the names of people in archival materials. When possible, they used VIAF, but other times they needed to create their own URI denoting an individual.

There is a level of rigor involved in creating the data model, and there may be many discussions regarding semantics. For example, what is a creator? Or, when is a term intended to be an index term as opposed reference. When does one term in one vocabulary equal a different term in a different vocabulary?

Balance the creation of your own vocabulary with the need to speak the language of others using their vocabulary.

Consider "fixing" the data as it comes in or goes out because it might not be consistent nor thorough.

Provenance is an issue. People — especially scholars — will want to know where the LOD came from and whether or not it is authoritative. How to solve or address this problem? The jury is still out on this one.

Creating and maintaining LOD is difficult because it requires the skills of a number of different types of people. Computer programmers. Database designers. Subject experts. Metadata specialists. Archivists. Etc. A team is all but necessary.

Objectives

[Management, access, and use and linked data affordances]

Overview of linked data concepts and vocabulary

Linked Data is a process for systematically and methodically exposing metadata on the Web. In many ways, it is the re-articulation of a thing called the Semantic Web first outlined more than a decade ago. Linked Data (and the Semantic Web) are efforts to increase the "sphere of knowledge" through the use of computer technology.

Increasingly you will hear of of linked data being qualified as "linked open data". The "open" qualifier alludes to the important distinctions between truly free data/information and licensed data/information which comes with some strings attached. Truly "open" linked data comes with no licensing restrictions.

When you hear of linked data and the Semantic Web, the next thing you often hear is "RDF" or "Resource Description Framework". First and foremost, RDF is a way of representing knowledge. It does this through the use of assertions (think, "sentences") with only three parts: 1) a subject, 2) a predicate, and 3) an object. Put together, these three things create things called "triples". The subject of each assertion is expected to be a Universal Resource Identifier (or URI, but think URL), and this URI is expected to represent a thing -- anything. (Really, anything.) The predicate is some sort of relationship such as equals or is a sub-part of or contains or is a description of, or is the name of, etc. Predicates are the vocabulary of linked data, and you will find an abundance of vocabularies from which to choose when creating Linked Data. Finally, objects come in two forms: 1) more URIs (pointers to things) or literal values such the names of people, places, or things. Examples of literals include “Lancaster, PA”, “Thomas Jefferson”, or “Musée d’Orsay”.

RDF is not to be confused with RDF/XML or another other type of RDF “serialization”. Remember, RDF describes triples, but it does not specify how the triples are express or written down. On the other hand, RDF/XML is an XML syntax for expressing RDF. Some people think RDF/XML is too complicated and too verbose. Consequently, other serializations have manifested themselves including N3 and Turtle.

Brief overview of the history of LOD-LAM

Examples

Linked Data Today

Projects

ReLoad

The ReLoad project (Repository for Linked open archival data) will foster experimentation with the technology and methods of linked open data for archival resources. Its goal is the creation of a web of linked archival data.

LOD-LAM, which is an acronym for Linked Open Data for Libraries, Archives and Museums, is an umbrella term for the community and active projects in this area.

The first experimental phase will make use of W3C semantic web standards, mash-up techniques, software for linking and for defining the semantics of the data in the selected databases.

The archives that have made portions of their institutions' data and databases openly available for this project are the Central State Archive, and the Cultural Heritage Institute of Emilia Romagna Region. These will be used to test methodologies to expose the resources as linked open data.

http://labs.regesta.com/progettoReload/en

OAD Vocabulary Specification 1.2 - http://labs.regesta.com/progettoReload/wp-content/uploads/2013/08/oadNew.html

Reload project (http://labs.regesta.com/progettoReload/), an Italian experimentation started in 2012 and supported by Central State Archive, Cultural Heritage Institute of Emilia Romagna Region and regesta.exe, published the first version of OAD ontology (ontology for archival description) based on ISAD (G) standard and EAD schema, in 2013. The project team define a specific ontology for archival description domain because there isn't yet something useful to describe all significant classes and properties necessary in archival description and we integrated OAD with other "Lightweight ontologies" (like foaf or dublin core) to encode the most common metadata. Within Reload project, to describe authority records in Linked Data we use EAC-CPF ontology, published in 2012 by Cultural Heritage Institute of Emilia Romagna Region.

Reload project was presented during the last LODLAM Challenge in Montréal (http://summit2013.lodlam.net/2012/12/21/announcing-heat-1-lodlam-challenge-finalists/).

You are welcome in any integration and suggestion you'd like to do in this work.
Reload project website: http://labs.regesta.com/progettoReload/

--
Silvia Mazzini
regesta.exe srl
via Monte Zebio 19 - 00195 Roma
vox +39 0637501056 | +39 3477357729
fax +39 0637410460
smazzini@regesta.com | www.regesta.com | @regesta_sm

Google (and Facebook) knowledge graphs

OpenCat

Another common theme / application demonstrated at the conference were variations of the venerable library catalog. OpenCat, presented by Agnes Simon (Bibliothéque Nationale de France), was an additional example of this trend. Combining authority data (available as RDF) provided by the National Library of France with works of a second library (Fresnes Public Library), the OpenCat prototype provides quite an interesting interface to library holdings. --http://demo.cubicweb.org/opencatfresnes/

Newspaper Clippings Archives

On Jan 8, 2014, at 12:05 PM, Neubert Joachim <J.Neubert@zbw.eu>
wrote:

Thank you for your report on SWIB13! I'm glad you enjoyed the
conference and your stay in Hamburg.

I wanted to get in touch with you because you mentioned in your
blog that you are working on a book about LOD in archives.
Perhaps, in your research for that, you came across
press/newspaper clippings archives.

As you may know, I've published the persons and company part of
the 20th Century Press Archives (http://zbw.eu/beta/p20) as a
linked data application. It uses RDFa and OAI-ORE extensively to
give every dossier, every article and every page a citable URI,
and on the other hand consumes linked data from various linked
data sources to enrich the web pages and to provide context to
the rather plain scanned article images.

I wonder if there are other archives, and particular newspaper
archives, out there which do similar things, and would be very
happy about hints.

http://challenge.semanticweb.org/submissions/swc2010_submission_6.pdf
http://elag2011.techlib.cz/files/download/id/45/drawing-context-from-the-linked-data-web-the-20th-century-press-archives-neubert.pdf
http://www.w3.org/2005/Incubator/lld/wiki/Use_Case_Publishing_20th_Century_Press_Archives

--
Joachim Neubert
ZBW – German National Library of Economics
Leibniz Information Centre for Economics
Neuer Jungfernstieg 21
20354 Hamburg

LOCAH Project

Mimas and UKOLN worked together on an exciting JISC funded project to make Archives Hub data available as structured Linked Data, for the benefit of education and research. We worked in partnership with Eduserv, Talis and OCLC, leading experts within their fields. The aim was put archival and bibliographic data at the heart of the Linked Data Web, enabling new links to be made between diverse content sources and enabling the free and flexible exploration of data so that researchers can make new connections between subjects, people, organisations and places to reveal more about our history and society. --http://archiveshub.ac.uk/locah/

Linking Lives

Linking Lives is exploring ways to present Linked Data. We aim to show that archives can benefit from being presented as a part of the diverse data sources on the Web to create full biographical pictures, enabling researchers to make connections between people and events.

Linking Lives builds upon the Locah project. Locah was a JISC-funded project to expose the Archives Hub descriptions as Linked Data. --http://archiveshub.ac.uk/linkinglives/

Linked Archives Hub Test Dataset

The dataset describes archives held by UK institutions. The data is derived from a sample of the archival finding aids held by the UK Archives Hub. --http://data.archiveshub.ac.uk

Trends in LOD-LAM

- Mash ups

- Harvesting along side other protocols
- Increased interest
- Increased number of RDF serializations
- Governments making their content available
- Using them to enhance online catalogs
- Creating timelines
- Creating “named graphs”
- Increased number of programming toolkits
- Emphasis on “open” linked data and linked data in museums and archives
- Making RDF dumps available
- Interest in schema.org

With great interest I read the Spring/Summer issue of Information Standards Quarterly where there were a number of articles pertaining to linked open data in cultural heritage institutions. [0] Of particular interest to me where the various loosely enumerated challenges of linked open data. Some of them included:

- the apparent Tower Of Babel when it comes to vocabularies used to describe content, and the same time we need to have “ontology mindfulness”.
- dirty, inconsistent, or wide varieties of data integrity
- persistent URIs
- the “chicken & egg” problem of why linked data if there is no killer application

Getting Started: Strategies and Steps

Defining your strategy

Linked data represents a modern way of making your archival descriptions accessible to the wider world. In that light, it represents a different way of doing things but not necessary a different what of doing things. You will still be doing inventory. You will still be curating collections. You will still be prioritizing what goes and what stays.

On the other hand, linked data changes the way your descriptions get expressed and distributed. It is a lot like taking a trip across country. The goal was always to get to the coast to see the ocean, but instead of walking, going by stage coach, taking a train, or driving a car, you will be flying. Along the way you may visit a few cities and have a few layovers. Bad weather may even get in the way, but sooner or later you will get to your destination. Take a deep breath. Understand that the process will be one of learning, and that learning will be applicable in other aspects of your work. The result will be two-fold. First, a greater number of people will have access to your collections, and consequently, more people will will be using your collections.

With this in mind, articulate some goals — broad targets of things you would like to accomplish. Some of them might include:

- making your archival collections more widely accessible
- working with others to build virtual collections of like topics or formats
- incorporating your archival descriptions into public spaces like Wikipedia
- integrating your collections into local teaching, learning, and research activities
- increasing the awareness of your archive to benefactors
- increasing the computer technology skills of fellow archivists

How might you go about accomplishing these goals? What are your objectives? (What method of transportation are you going to use to get where you are going?) How am I going to measure success? In other words, you will need to create an plan, and each item in the plan answers a simple question — Who is going to do what by when? In other word, what people will be responsible for accomplishing the particular objective. Exactly what will they be doing, and by what time will they have it accomplished. Each of these components are described in greater detail below

Who

It is quite unlikely your linked data goals and objectives will be accomplished by a single person. Instead it will most likely required a team of people. These people do not necessarily need to working in the same physical location, but they will require a diverse set of skills. Some of them include, and each plays a key, indispensable role:

content specialists - These are the people who understand the “aboutness” of a particular collection. These are the people who understand and can thoroughly articulate the significance of a collection. They know how and why particular things belong in a collection. They are able to answer questions about the collection as all as tell stories against it.

metadata specialists - These are people who understand data about data. Not only do they understand the principles of controlled vocabularies and authority lists, but they are also familiar with a wide variety of such lists, specifically as they are represented on the Web. In linked data there are fewer descriptive cataloging “rules”. Nevertheless, the way the ontologies of linked data can be used need to be interpreted, and this interpretation needs to be consistent. Metadata specialists understand these principles.

computer technologists - Not only are these the people who have a fundamental understanding of what computer can and cannot do, but they also know how to put this understanding into practice. At the very least, the computer technologists need to understand a myriad of data structures and how to convert them into different data structures. Converting MARC 21 into MARCXML. Transforming EAD into HTML. Reporting against a relational database to create serialized RDF. These tasks required computer programming skills, but not necessarily any one in particular. Any modern programming language (Java, PHP, Python, Ruby, etc.) includes the necessary function to complete the tasks.

What

The what of your objectives are not so much identified with nouns as they are action verbs, such as: write, evaluate, implement, examine, purchase, hire, prioritize, list, delete, acquire, discuss, share, find, compare & contrast, stop, start, complete, continue, describe, edit, updated, create, purchase, upgrade, etc. The what of your objective is in the doing.

When

The say, “Work expands to fill the available space.” If this is true, and no deadlines are articulated for each objective, then the allotted amount of time for any given task is all but infinite, but this it not true. Time is one of the most limited resources you have. When thinking about a given objective, ask yourself how much time you think it will take, multiply the time by one and a half. Ask yourself when the task can begin and document the beginning point as well as the estimated ending point. Do this all of your objectives and the result will be a Gantt chart. It will now be easy to look at the chart on a regular basis to see who things are progressing.

[Articulate goals, objectives, and metrics to measure success.]

Is your archival description LOD-ready?

Identify building blocks

Readiness

[Making small changes in practice to make your description LOD-ready.]

What you can do now if you have

Nothing - consider using RDFa

EAD

If you have used EAD to describe your collections, then you can easily make your descriptions available as valid linked data, but the result will be less than optimal. This is true not for a lack of technology but rather from the inherent purpose and structure of EAD files.

A few years ago an organisation in the United Kingdom called the Archive's Hub was funded by a granting agency called JISC to explore the publishing of archival descriptions as linked data. One of the outcomes of this effort was the creation of an XSL stylesheet transforming EAD into RDF/XML. The terms used in the stylesheet originate from quite a number of standardized, widely accepted ontologies, and with only the tiniest bit configuration / customization the stylesheet can transform a generic EAD file into valid RDF/XML. The resulting XML files can then be made available on a Web server or incorporated into a triple store. This goes a long way to publishing archival descriptions as linked data. The only additional things needed are a transformation of EAD into HTML and the configuration of a Web server to do content-negotiation between the XML and HTML.

For the smaller archive with only a few hundred EAD files whose content does not change very quickly, this is a simple, feasible, and practical solution to publishing archival descriptions as linked data. With the exception of doing some content-negotiation, this solution does not require any computer technology that is not already being used in archives, and it only requires a few small tweaks to a given workflow:

1. implement a content-negotiation solution
2. edit EAD file
3. transform EAD into RDF/XML
4. transform EAD into HTML
5. save the resulting XML and HTML files on a Web server
6. go to step #2

On the other hand an EAD file is the combination of a narrative description with a hierarchal inventory list, and this data structure does not lend itself very well to the triples of linked data. For example, EAD headers are full of controlled vocabularies terms but there is no way to link these terms with specific inventory items. This is because the vocabulary terms are expected to describe the collection as a whole, not individual things. This problem could be overcome if each individual component of the EAD were associated with controlled vocabulary terms, but this would significantly increase the amount of work needed to create the EAD files in the first place.

The common practice of using literals ("strings") to denote the names of people, places, and things in EAD files would also need to be changed in order to fully realize the vision of linked data. Specifically, it would be necessary for archivists to supplement their EAD files with commonly used URIs denoting subject headings and named authorities. These URIs could be inserted into id attributes throughout an EAD file, and the resulting RDF would be more linkable, but the labor to do so would increase, especially since many of the named authorities will not exist in standardized authority lists.

Despite these short comings, transforming EAD files into some sort of serialized RDF goes a long way towards publishing archival descriptions as linked data. This particular process is a good beginning and outputs valid information, just information that is not as accurate as possible. This process lends itself to iterative improvements, and outputting something is better than outputting nothing. But this particular proces is not for everybody. The archive whose content changes quickly, the archive with copious numbers of collections, or the archive wishing to publish the most accurate linked data possible will probably not want to use EAD files as the root of their publishing system. Instead some sort of database application is probably the best solution.

EAC-CPF

Encoded Archival Context for Corporate Bodies, Persons, and Families (EAC-CPF) goes a long way to implementing a named authority database that could be linked from archival descriptions. These XML files could easily be transformed into serialized RDF and therefore linked data. The resulting URIs could then be incorporated into archival descriptions making them richer and complete.

For example the FindAndConnect site in Australia uses EAC-CPF under the hood to disseminate information about people in its collection -- http://www.findandconnect.gov.au. Similarly, “SNAC aims to not only make the [EAC-CPF] records more easily discovered and accessed but also, and at the same time, build an unprecedented resource that provides access to the socio-historical contexts (which includes people, families, and corporate bodies) in which the records were created” -- http://socialarchive.iath.virginia.edu More than a thousand EAC-CPF records are available from the RAMP project -- http://demo.rampeditor.info/export.php

MARC

In some ways MARC lends it self very well to being published via linked data, but in the long run it is not really a feasible data structure.

Converting MARC into serialized RDF through XSLT is at least a two step process. The first step is to convert MARC into MARCXML. This can be done with any number of scripting languages and toolboxes. The second step is to use a stylesheet such as the one provided by the Library of Congress to transform the MARCXML into RDF/XML. From there a person could save the resulting XML files on a Web server, enhance access via content negotiation, and called it linked data.

Unfortunately, this particular approach has a number of drawbacks. First and foremost, the MARC format had no place to denote URIs; MARC records are made up almost entirely of literals. Sure, URIs can be constructed from various control numbers, but things like authors, titles, subject headings, and added entries will most certainly be strings (“Mark Twain”, “Adventures of Huckleberry Finn”, “Bildungsroman”, or “Samuel Clemans”), not URIs. This issue can be overcome if the MARCXML were first converted into MODS and URIs were inserted into id or xlink attributes of bibliographic elements, but this is extra work. If an archive were to take this approach, then it would also behoove them to use MODS as their data structure of choice, not MARC. Continually converting from MARC to MARCXML to MODS would be expensive in terms of time. Moreover, with each new conversion the URIs from previous iterations would need to be re-created.

METS and MODS

If you have archival descriptions in either of the METS or MODS formats, then transforming them into RDF is as far away as your XSLT processor and a content negotiation implementation. As of this writing there do not seem to be any METS to RDF stylesheets, but there are a couple stylesheets for MODS. The biggest issue with these sorts of implementations are the URIs. It will be necessary for archivists to include URIs into as many MODS id or xlink attributes as possible. The same thing holds true for METS files except the id attribute is not designed to hold external identifiers and therefore not a valid placeholder for URIs.

Databases

Publishing linked data through XML transformation is functional but not optimal. Publishing linked data from a database comes closer to the ideal but requires a greater amount of technical computer infrastructure and expertise.

Databases -- specifically, relational databases -- are the current best practice for organizing data. As you may or may not know, relational databases are made up of many tables of data joined with keys. For example, a book may be assigned a unique identifier. The book has many characteristics such as a title, number of pages, size, descriptive note, etc. Some of the characteristics are shared by other books, like authors and subjects. In a relational database these shared characteristics would be saved in additional tables, and they would be joined to a specific book through the use of unique identifiers (keys). Given this sort of data structure, reports can be created from the database describing its content. Similarly, queries can be applied against the database to uncover relationships that may not be apparent at first glance or buried in reports. The power of relational databases lay in the use of keys to make relationships between rows in one table and rows in other tables.

Not coincidently, this is very much the way linked data is expected to be implemented. In the linked data world, the subjects of triples are URIs (think database keys). Each URI is associated with one or more predicates (think the characteristics in the book example). Each triple then has an object, and these objects take the form of literals or other URIs. In the book example, the object could be “Adventures Of Huckleberry Finn” or a URI pointing to Mark Twain. The reports of relational databases are analogous to RDF serializations, and SQL (the relational database query language) is analogous to SPARQL, the query language of RDF triple stores. Because of the close similarity between well-designed relational databases and linked data principles, the publishing of linked data directly from relational databases makes whole lot of sense, but the process requires the combined time and skills of a number of different people: content specialists, database designers, and computer programmers. Consequently, the process of publishing linked data from relational databases may be optimal, but it is more expensive.

Thankfully, most archivists probably use some sort of database to manage their collections and create their finding aids. Moreover, archivists probably use one of three or four tools for this purpose: Archivist's Toolkit, Archon, ArchivesSpace, or PastPerfect. Each of these systems have a relational database at their heart. Reports could be written against the underlying databases to generate serialized RDF and thus begin the process of publishing linked data. Doing this from scratch would be difficult, as well as inefficient because many people would be starting out with the same database structure but creating a multitude of varying outputs. Consequently, there are two alternatives. The first is to use a generic database application to RDF publishing platform called D2RQ. The second is for the community to join together and create a holistic RDF publishing system based on the database(s) used in archives.

D2RQ is a wonderful software system. It is supported, well-documented, executable on just about any computing platform, open source, focused, functional, and at the same time does not try to be all things to all people. Using D2RQ it is more than possible to quickly and easily publish a well-designed relational database as RDF. The process is relatively simple:

1. download the software
   |
2. use a command-line utility to map the database structure to a configuration file

3. season the configuration file to taste

4. run the D2RQ server using the configuration file as input thus allowing people or RDF user-agents to search and browse the database using linked data principles

5. alternatively, dump the contents of the database to an RDF serialization and upload the result into your favorite RDF triple store

The downside of D2RQ is its generic nature. It will create an RDF ontology whose terms correspond to the names of database fields. These field names do not map to widely accepted ontologies and therefore will not interact well with communities outside the ones using a specific database structure. Still, the use of D2RQ is quick, easy, and accurate.

The second alternative to using databases of archival content to published linked data requires community effort and coordination. The databases of Archivist's Toolkit, Archon, ArchivesSpace, or Past Perfect could be assumed. The community could then get together and create and decide on an RDF ontology to use for archival descriptions. The database structure(s) could then be mapped to this ontology. Next, programs could be written against the database(s) to create serialized RDF thus beginning the process of publishing linked data. Once that was complete, the archival community would need to come together again to ensure it uses as many shared URIs as possible thus creating the most functional sets of linked data. This second alternative requires a significant amount of community involvement and wide-spread education. It represents a never-ending process.

On Your Way: Next Steps

Integration into daily practice

Three Cs: Cleanup, Conversion, Consistency

The article entitled Recipes for Enhancing Digital Collections with Linked Data by Thomas Johnson and Karen Estlund (http://journal.code4lib.org/articles/9214) outlines a number of ways of cleaning up data in content management systems by way of RDF statements.

clean up steps include:

1. Remove noise
2. Normalize presentation
3. Assign URIs for curation objects
4. Map legacy elements to Linked Data vocabularies

As stated by Hillman, the process of moving to linked data is The key to this augmentation process involves changing the basic metadata unit from “record” to “statement.” — http://dcpapers.dublincore.org/pubs/article/view/770/766

Problems with data, again from hillman an:

1. missing data – metadata elements not present in supplied metadata
2. incorrect data – metadata values not con- forming to standard element use
3. confusing data – multiple values crammed into a single metadata element, embedded html tags, etc.
4. insufficient data – e.g., no indication of controlled vocabularies used

Safe transformations include:

1. remove “noise” – a partial solution to the “incorrect data” problem. For example, we remove metadata with no information value, such as empty metadata elements, metadata elements with values such as “unknown” or “n/a” or consisting entirely of dashes or other punctuation.

2. detect and identify controlled vocabular- ies in use whenever possible – a partial solution to the “insufficient data” prob- lem. For example, the DCMIType encod- ing scheme is applied to DC “Type” elements when their value is one of the allowed DCMITypes [10]. This works well for small controlled vocabularies; however, it does not scale well to large vocabularies such as LCSH.

3. normalize metadata presentation – clean up the values: remove double XML en-codings (“&amp;lt;” becomes “&lt;”), extra whitespace (a tab followed by five spaces becomes a single space), etc.

Creating and maintaining metadata is a never-ending process. The items being described can always use elaboration. Collections may increase is size. Rights applied against content may change. Things become digitized, or digitized things are migrated from one format to another. Because of these sorts of things and many others, cleanup, conversion, and consistency are something every metadata specialist needs to keep in mind.

Cleanup, conversion, and consistency means many things. Does all of your metadata use the same set of one or more vocabularies? Are things spelled correctly? Maybe you used abbreviations in one document but spelled things out in another? Have you migrated your JPEG images to JPEG2000 or TIFF formats? Maybe the EAD DTD has been updated, and you want (need) to migrate your finding aids from one XML format to another? Do all of your finding aids exhibit the same level of detail; are some “thinner” than others? Have you used one form of a person’s name in one document but used another form in a different document? The answers to these sorts of questions point to the need for cleanup, conversion, and consistency.

Tools

- Fusion Tables (http://www.google.com/drive/apps.html) - Bust your data out of its silo! Combine it with other data on the web. Collaborate, visualize and share.

- OpenRefine (https://github.com/OpenRefine/) - OpenRefine is a free, open source power tool for working with messy data and improving it

- cURL - curl -L -H ‘Accept: application/rdf+xml’ http://infomotions.com/sandbox/liam/id/ctumarc15567

Looking Ahead: Advanced Tools and Visualizations

Tools for archivists (data preparation, cleanup, management)

What's available now

- Bibframe (http://bibframe.org) - The Bibliographic Framework Initiative (BIBFRAME) is an undertaking by the Library of Congress and the community to better accommodate future needs of the library community. A major focus of the initiative will be to determine a transition path for the MARC 21 exchange format to more Web based, Linked Data standards. Zepheira and The Library of Congress are working together to develop a Linked Data model, vocabulary and enabling tools / services for supporting this Initiative.

- ckan (http://ckan.org) - The open source data portal software

- CouchDB (http://couchdb.apache.org) - CouchDB is a database that completely embraces the web. Store your data with JSON documents. Access your documents with your web browser, via HTTP. Query, combine, and transform your documents with JavaScript. CouchDB works well with modern web and mobile apps. You can even serve web apps directly out of CouchDB. And you can distribute your data, or your apps, efficiently using CouchDB's incremental replication. CouchDB supports master-master setups with automatic conflict detection.

- Curl (http://curl.haxx.se) - curl is a command line tool for transferring data with URL syntax, supporting DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, Telnet and TFTP. curl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, kerberos...), file transfer resume, proxy tunneling and a busload of other useful tricks.

- D2RQ (http://d2rq.org) - The D2RQ Platform is a system for accessing relational databases as virtual, read-only RDF graphs. It offers RDF-based access to the content of relational databases without having to replicate it into an RDF store. Using D2RQ you can: query a non-RDF database using SPARQL, access the content of the database as Linked Data over the Web, create custom dumps of the database in RDF formats for loading into an RDF store, access information in a non-RDF database using the Apache Jena API

- Datahub (http://datahub.io/) - the free, powerful data management platform from the Open Knowledge Foundation

- Disco - Hyperdata Browser (http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/disco/) - The Disco - Hyperdata Browser is a simple browser for navigating the Semantic Web as an unbound set of data sources. The browser renders all information, that it can find on the Semantic Web about a specific resource, as an HTML page. This resource description contains hyperlinks that allow you to navigate between resources. While you move from resource to resource, the browser dynamically retrieves information by dereferencing HTTP URIs and by following rdfs:seeAlso links.

- ead2rdf (http://data.archiveshub.ac.uk/xslt/ead2rdf.xsl) - The "transform" process is currently performed using XSLT to read an EAD XML document and output RDF/XML, and the current version of the stylesheet is now available:

- eaditor (https://github.com/ewg118/eaditor) - EADitor is an XForms framework for the creation and editing of Encoded Archival Description (EAD) finding aids using Orbeon, an enterprise-level XForms Java application, which runs in Apache Tomcat.

- Fusion Tables (http://www.google.com/drive/apps.html) - Bust your data out of its silo! Combine it with other data on the web. Collaborate, visualize and share.

- Linked Data Tools (http://linkeddata.org/tools) -

- Linked Media Framework (https://code.google.com/p/lmf/) - The Linked Media Framework is an easy-to-setup server application that bundles together some key open source projects to offer some advanced services for linked media management.

- oai2lod (https://github.com/behas/oai2lod) - exposes OAI-PMH data sources as Linked Data

- OpenLink Data Explorer Extension (http://ode.openlinksw.com) - The OpenLink Data Explorer (ODE) is a browser extension (currently available for Firefox, Safari, Chrome, Opera, and Internet Explorer with additional browser support to follow) that adds a new option to the realm of Web User Agent functionality, in the form of new menu options for viewing Data Sources associated with Web Pages.

- OpenRefine (https://github.com/OpenRefine/) - OpenRefine is a free, open source power tool for working with messy data and improving it

- Perl and RDF (http://www.perlrdf.org) - The Perl RDF project hopes to address these issues:, publish an official API for storage, parsing and serializing modules, produce a set of base classes for representing common RDF objects such as statements and nodes (resources, literals, blank nodes), produce patches to existing RDF tools to support these APIs, subclassing where appropriate, produce a test suite for storage, parsing, serializing, statement and node classes.

- Perl-SPARQL-client-library (https://github.com/swh/Perl-SPARQL-client-library) - A simple Perl library for accessing SPARQL endpoints.

- Protégé (http://protege.stanford.edu) - Protégé is a free, open source ontology editor and knowledge-base framework The Protégé platform supports modeling ontologies via a web client or a desktop client. Protégé ontologies can be developed in a variety of formats including OWL, RDF(S), and XML Schema Protégé is based on Java, is extensible, and provides a plug-and-play environment that makes it a flexible base for rapid prototyping and application development.

- RDFImportersAndAdapters (http://www.w3.org/wiki/RDFImportersAndAdapters) - Tools and applications that can convert from other data and file formats to RDF.

- Semantic Web Development Tools (http://www.w3.org/2001/sw/wiki/Tools) - This Wiki contains a collection of tool references that can help in developing Semantic Web applications. These include complete development environments, editors, libraries or modules for various programming languages, specialized browsers, etc. The goal is to list such tools and not Semantic Web applications in general (the interested reader may consider looking at the W3C SW Use Case Collection for those.)

- Sematic Web Client Library (http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/semwebclient/) - The Sematic Web Client Library represents the complete Semantic Web as a single RDF graph. The library enables applications to query this global graph using SPARQL- and find(SPO) queries. To answer queries, the library dynamically retrieves information from the Semantic Web by dereferencing HTTP URIs, by following rdfs:seeAlso links, and by querying the Sindice search engine. The library is written in Java and is based on the Jena framework.

- SparqlImplementations (http://www.w3.org/wiki/SparqlImplementations) - This page lists some implementations of SPARQL, a query language and protocol for RDF acccess released by the W3C RDF Data Access Working Group - DAWG.

- Tableau Public (http://www.tableausoftware.com/public) - With Tableau Public you can create interactive graphs, dashboards, maps and tables from virtually any data and embed them on your website or blog in minutes.

- Tabulator (http://www.w3.org/2005/ajar/tab) - The Tabulator project is a generic data browser and editor. Using outline and table modes, it provides a way to browse RDF data on the web. RDF is the standard for inter-application data exchange.

- TemaTres (http://www.vocabularyserver.com) - The open source way to manage formal representations of knowledge

- VirtuosoUniversalServer (http://www.w3.org/wiki/VirtuosoUniversalServer) - OpenLink Virtuoso is a multi-purpose and multi-protocol (Hybrid) Data Server from OpenLink Software that includes SQL Object-Relational, RDF, XML, and Free Text data management, alongside Web Application (HTTP, SOAP, WebDAV), SyncML, and Discussion Server functionality, in a single server.

- W3C RDF Validation Service (http://www.w3.org/RDF/Validator/) - Enter a URI or paste an RDF/XML document into the text field above. A 3-tuple (triple) representation of the corresponding data model as well as an optional graphical visualization of the data model will be displayed.

Gaps: What is needed

There needs to be easy to use tools to find URIs and insert them in to archival descriptions. One such tool is called lobid:

> In “From strings to things: A linked data API for library hackers and Web developers” Fabian Steeg and Pascal Christoph (HBZ) described an interface allowing librarians to determine the URIs of people, places, and things for library catalog records. “How can we benefit from linked data without being linked data experts? We want to pub Web developers into focus using JSON for HTTP.” There are few hacks illustrating some of their work on Github in the lobid repository. --https://github.com/lobid

Another example would be an interface to the varius linked data sets available from the Library of Congress. --http://id.loc.gov

Tools for users: visualizations, interfaces, etc.

What's available now

- D3.js (http://d3js.org) - D3.js is a JavaScript library for manipulating documents based on data. D3 helps you bring data to life using HTML, SVG and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

- Gephi (http://gephi.org) - Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs.

- Tableau Public (http://www.tableausoftware.com/public) - With Tableau Public you can create interactive graphs, dashboards, maps and tables from virtually any data and embed them on your website or blog in minutes.

Gaps: What is needed

Listed in no priority order, some of the things needed, include:

- hands-on training
- desktop tools enabling people or machines to associate strings with URIs
- a simple RDF statement editor
- the killer app / additional demonstration applications
- a conceptional shift from document to statement

Appendix: Tools

- "4store - Scalable RDF Storage." Accessed November 12, 2013. http://4store.org/.

- "Apache Jena - Home." Accessed November 11, 2013. http://jena.apache.org/.

- "Behas/oai2lod · GitHub." Accessed November 3, 2013. https://github.com/behas/oai2lod.

- "BIBFRAME.ORG :: Bibliographic Framework Initiative - Overview." Accessed November 3, 2013. http://bibframe.org/.

- "Ckan - The Open Source Data Portal Software." Accessed November 3, 2013. http://ckan.org/.

- "Community | Tableau Public." Accessed November 3, 2013. http://www.tableausoftware.com/public/community.

- "ConverterToRdf - W3C Wiki." Accessed November 11, 2013. http://www.w3.org/wiki/ConverterToRdf.

- "Curl and Libcurl." Accessed November 3, 2013. http://curl.haxx.se/.
- "D2R Server | The D2RQ Platform." Accessed November 15, 2013. http://d2rq.org/d2r-server.

- "Disco Hyperdata Browser." Accessed November 3, 2013. http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/disco/.

- "Ead2rdf." Accessed November 3, 2013. http://data.archiveshub.ac.uk/xslt/ead2rdf.xsl.

- “Ewg118/eaditor · GitHub.” Accessed November 3, 2013. https://github.com/ewg118/eaditor.
- “Google Drive.” Accessed November 3, 2013. http://www.google.com/drive/apps.html.
- Library, The standard EAC-CPF is maintained by the Society of American Archivists in partnership with the Berlin State. “Society of American Archivists and the Berlin State Library.” Accessed January 1, 2014. http://eac.staatsbibliothek-berlin.de/.
- “Lmf - Linked Media Framework - Google Project Hosting.” Accessed November 3, 2013. https://code.google.com/p/lmf/.
- “OpenLink Data Explorer Extension.” Accessed November 3, 2013. http://ode.openlinksw.com/.
- “openRDF.org: Home.” Accessed November 12, 2013. http://www.openrdf.org/.
- “OpenRefine (OpenRefine) · GitHub.” Accessed November 3, 2013. https://github.com/OpenRefine/.
- “Parrot, a RIF and OWL Documentation Service.” Accessed November 11, 2013. http://ontorule-project.eu/parrot/parrot.
- “RDF2RDF - Converts RDF from Any Format to Any.” Accessed December 5, 2013. http://www.l3s.de/~minack/rdf2rdf/.
- “RDFImportersAndAdapters - W3C Wiki.” Accessed November 3, 2013. http://www.w3.org/wiki/RDFImportersAndAdapters.
- “RDFizers - SIMILE.” Accessed November 11, 2013. http://simile.mit.edu/wiki/RDFizers.

- “Semantic Web Client Library.” Accessed November 3, 2013. http://wifo5-03.informatik.uni-mannheim.de/bizer/ng4j/semwebclient/.
- “SIMILE Widgets | Exhibit.” Accessed November 11, 2013. http://www.simile-widgets.org/exhibit/.
- “SparqlImplementations - W3C Wiki.” Accessed November 3, 2013. http://www.w3.org/wiki/SparqlImplementations.
- “swh/Perl-SPARQL-client-library · GitHub.” Accessed November 3, 2013. https://github.com/swh/Perl-SPARQL-client-library.
- “Tabulator: Generic Data Browser.” Accessed November 3, 2013. http://www.w3.org/2005/ajar/tab.
- “TaskForces/CommunityProjects/LinkingOpenData/SemWebClients - W3C Wiki.” Accessed November 5, 2013. http://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/SemWebClients.
- “TemaTres Controlled Vocabulary Server.” Accessed November 3, 2013. http://www.vocabularyserver.com/.
- “The D2RQ Platform – Accessing Relational Databases as Virtual RDF Graphs.” Accessed November 3, 2013. http://d2rq.org/.
- “The Protégé Ontology Editor and Knowledge Acquisition System.” Accessed November 3, 2013. http://protege.stanford.edu/.
- “Tools - Semantic Web Standards.” Accessed November 3, 2013. http://www.w3.org/2001/sw/wiki/Tools.
- “Tools | Linked Data - Connect Distributed Data Across the Web.” Accessed November 3, 2013. http://linkeddata.org/tools.

- “Vapour, a Linked Data Validator.” Accessed November 11, 2013. http://validator.linkeddata.org/vapour.

- “VirtuosoUniversalServer - W3C Wiki.” Accessed November 3, 2013. http://www.w3.org/wiki/VirtuosoUniversalServer.

- “W3C RDF Validation Service.” Accessed November 3, 2013. http://www.w3.org/RDF/Validator/.

- “W3c/rdfvalidator-ng.” Accessed December 10, 2013. https://github.com/w3c/rdfvalidator-ng.

- “Working with RDF with Perl.” Accessed November 3, 2013. http://www.perlrdf.org/.

Appendix: Data sets

- "(LOV) Linked Open Vocabularies." Accessed November 3, 2013. http://lov.okfn.org/dataset/lov/.

- "Data Sets & Services." Accessed November 3, 2013. http://www.oclc.org/data/data-sets-services.en.html.

- "Data.gov.uk." Accessed November 3, 2013. http://data.gov.uk/.

- "Freebase." Accessed November 3, 2013. http://www.freebase.com/.

- "GeoKnow/LinkedGeoData · GitHub." Accessed November 3, 2013. https://github.com/GeoKnow/LinkedGeoData.

- "GeoNames." Accessed November 3, 2013. http://www.geonames.org/.

- "Getty Union List of Artist Names (Research at the Getty)." Accessed November 3, 2013. http://www.getty.edu/research/tools/vocabularies/ulan/.

- "Home - LC Linked Data Service (Library of Congress)." Accessed November 3, 2013. http://id.loc.gov/.

- "Home | Data.gov." Accessed November 3, 2013. http://www.data.gov/.
- "ISBNdb - A Unique Book & ISBN Database." Accessed November 3, 2013. http://isbndb.com/.

- "Linked Movie Data Base | Start Page." Accessed November 3, 2013. http://linkedmdb.org/.

- "MusicBrainz - The Open Music Encyclopedia." Accessed November 3, 2013. http://musicbrainz.org/.

- “New York Times - Linked Open Data.” Accessed November 3, 2013. http://data.nytimes.com/.

- “PELAGIOS: About PELAGIOS.” Accessed September 4, 2013. http://pelagios-project.blogspot.com/p/about-pelagios.html.

- “Start Page | D2R Server for the CIA Factbook.” Accessed November 3, 2013. http://wifo5-03.informatik.uni-mannheim.de/factbook/.

- “Start Page | D2R Server for the Gutenberg Project.” Accessed November 3, 2013. http://wifo5-03.informatik.uni-mannheim.de/gutendata/.

- “The Friend of a Friend (FOAF) Project | FOAF Project.” Accessed November 3, 2013. http://www.foaf-project.org/.

- “VIAF.” Accessed August 27, 2013. http://viaf.org/.

- “VoID - Semanticweb.org.” Accessed November 3, 2013. http://semanticweb.org/wiki/VoID.

- “Web Data Commons.” Accessed November 19, 2013. http://webdatacommons.org/.

- “Welcome - the Datahub.” Accessed August 14, 2013. http://datahub.io/.

- “Welcome to Open Library (Open Library).” Accessed November 3, 2013. https://openlibrary.org/.

- “Wiki.dbpedia.org : About.” Accessed November 3, 2013. http://dbpedia.org/About.

- “World Bank Linked Data.” Accessed November 3, 2013. http://worldbank.270a.info/.html.

Appendix: Further reading

This is a list of links and citations to get one started on Linked Open Data

- admin. “Barriers to Using EAD,” August 4, 2012. http://oclc.org/research/activities/eadtools.html.

- Becker, Christian, and Christian Bizer. “Exploring the Geospatial Semantic Web with DBpedia Mobile.” Web Semantics: Science, Services and Agents on the World Wide Web 7, no. 4 (December 2009): 278–286. doi:10.1016/j.websem.2009.09.004.

- Belleau, François, Marc-Alexandre Nolin, Nicole Tourigny, Philippe Rigault, and Jean Morissette. “Bio2RDF: Towards a Mashup to Build Bioinformatics Knowledge Systems.” Journal of Biomedical Informatics 41, no. 5 (October 2008): 706–716. doi:10.1016/j.jbi.2008.03.004.

- Berners-Lee, Tim. “Linked Data - Design Issues.” Accessed August 4, 2013. http://www.w3.org/DesignIssues/LinkedData.html.

- Berners-Lee, Tim, James Hendler, and Ora Lassila. “The Semantic Web.” Scientific American 284, no. 5 (May 2001): 34–43. doi:10.1038/scientificamerican0501-34.

- Bizer, Christian, Tom Heath, and Tim Berners-Lee. “Linked Data - The Story So Far:” International Journal on Semantic Web and Information Systems 5, no. 3 (33 2009): 1–22. doi:10.4018/jswis.2009081901.

- Carroll, Jeremy J., Christian Bizer, Pat Hayes, and Patrick Stickler. “Named Graphs.” Web Semantics: Science, Services and Agents on the World Wide Web 3, no. 4 (December 2005): 247–267. doi:10.1016/j.websem.2005.09.001.

- “Chem2bio2rdf - How to Publish Data Using D2R?” Accessed January 6, 2014. http://chem2bio2rdf.wikispaces.com/How+to+publish+data

+using+D2R%3F.

- "Content Negotiation." Wikipedia, the Free Encyclopedia, July 2, 2013. https://en.wikipedia.org/wiki/Content_negotiation.

- "Cool URIs for the Semantic Web." Accessed November 3, 2013. http://www.w3.org/TR/cooluris/.

- Correndo, Gianluca, Manuel Salvadores, Ian Millard, Hugh Glaser, and Nigel Shadbolt. "SPARQL Query Rewriting for Implementing Data Integration over Linked Data." 1. ACM Press, 2010. doi: 10.1145/1754239.1754244.

- David Beckett. "Turtle." Accessed August 6, 2013. http://www.w3.org/TR/2012/WD-turtle-20120710/.

- "Debugging Semantic Web Sites with cURL | Cygri's Notes on Web Data." Accessed November 3, 2013. http://richard.cyganiak.de/blog/2007/02/debugging-semantic-web-sites-with-curl/.

- Dunsire, Gordon, Corey Harper, Diane Hillmann, and Jon Phipps. "Linked Data Vocabulary Management: Infrastructure Support, Data Integration, and Interoperability." Information Standards Quarterly 24, no. 2/3 (2012): 4. doi:10.3789/isqv24n2-3.2012.02.

- Elliott, Thomas, Sebastian Heath, and John Muccigrosso. "Report on the Linked Ancient World Data Institute." Information Standards Quarterly 24, no. 2/3 (2012): 43. doi:10.3789/isqv24n2-3.2012.08.

- Fons, Ted, Jeff Penka, and Richard Wallis. "OCLC's Linked Data Initiative: Using Schema.org to Make Library Data Relevant on the Web." Information Standards Quarterly 24, no. 2/3 (2012): 29. doi: 10.3789/isqv24n2-3.2012.05.

- Hartig, Olaf. "Querying Trust in RDF Data with tSPARQL." In The Semantic Web: Research and Applications, edited by Lora Aroyo, Paolo

Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, 5554:5–20. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. http://www.springerlink.com/index/10.1007/978-3-642-02121-3_5.

- Hartig, Olaf, Christian Bizer, and Johann-Christoph Freytag. “Executing SPARQL Queries over the Web of Linked Data.” In The Semantic Web - ISWC 2009, edited by Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, 5823:293–309. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. http://www.springerlink.com/index/10.1007/978-3-642-04930-9_19.

- Heath, Tom, and Christian Bizer. “Linked Data: Evolving the Web into a Global Data Space.” Synthesis Lectures on the Semantic Web: Theory and Technology 1, no. 1 (February 9, 2011): 1–136. doi:10.2200/S00334ED1V01Y201102WBE001.

- Isaac, Antoine, Robina Clayphan, and Bernhard Haslhofer. “Europeana: Moving to Linked Open Data.” Information Standards Quarterly 24, no. 2/3 (2012): 34. doi:10.3789/isqv24n2-3.2012.06.

- Kobilarov, Georgi, Tom Scott, Yves Raimond, Silver Oliver, Chris Sizemore, Michael Smethurst, Christian Bizer, and Robert Lee. “Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections.” In The Semantic Web: Research and Applications, edited by Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Simperl, 5554:723–737. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. http://www.springerlink.com/index/10.1007/978-3-642-02121-3_53.

- LiAM. “LiAM: Linked Archival Metadata.” Accessed July 30, 2013. http://sites.tufts.edu/liam/.

- “Linked Data.” Wikipedia, the Free Encyclopedia, July 13, 2013. http://en.wikipedia.org/w/index.php?title=Linked_data&oldid=562554554.

- “Linked Data Glossary.” Accessed January 1, 2014. http://www.w3.org/TR/ld-glossary/.

- “Linked Open Data.” Europeana. Accessed September 12, 2013. http://pro.europeana.eu/web/guest;jsessionid=09A5D79E7474609AE246DF5C5A18DDD4.

- “Linked Open Data in Libraries, Archives, & Museums (Google Group).” Accessed August 6, 2013. https://groups.google.com/forum/#!forum/lod-lam.

- “Linking Lives | Using Linked Data to Create Biographical Resources.” Accessed August 16, 2013. http://archiveshub.ac.uk/linkinglives/.
- “LOCAH Linked Archives Hub Test Dataset.” Accessed August 6, 2013. http://data.archiveshub.ac.uk/.

- “LODLAM - Linked Open Data in Libraries, Archives & Museums.” Accessed August 6, 2013. http://lodlam.net/.

- “Notation3.” Wikipedia, the Free Encyclopedia, July 13, 2013. http://en.wikipedia.org/w/index.php?title=Notation3&oldid=541302540.

- “OWL 2 Web Ontology Language Primer.” Accessed August 14, 2013. http://www.w3.org/TR/2009/REC-owl2-primer-20091027/.

- Quilitz, Bastian, and Ulf Leser. “Querying Distributed RDF Data Sources with SPARQL.” In The Semantic Web: Research and Applications, edited by Sean Bechhofer, Manfred Hauswirth, Jörg Hoffmann, and Manolis Koubarakis, 5021:524–538. Berlin, Heidelberg: Springer Berlin Heidelberg. Accessed September 4, 2013. http://www.springerlink.com/index/10.1007/978-3-540-68234-9_39.

- “RDF/XML.” Wikipedia, the Free Encyclopedia, July 13, 2013. http://en.wikipedia.org/wiki/RDF/XML.

- “RDFa.” Wikipedia, the Free Encyclopedia, July 22, 2013. http://en.wikipedia.org/wiki/RDFa.

- “Semantic Web.” Wikipedia, the Free Encyclopedia, August 2, 2013. http://en.wikipedia.org/w/index.php?title=Semantic_Web&oldid=566813312.

- “SPARQL.” Wikipedia, the Free Encyclopedia, August 1, 2013. http://en.wikipedia.org/w/index.php?title=SPARQL&oldid=566718788.
- “SPARQL 1.1 Overview.” Accessed August 6, 2013. http://www.w3.org/TR/sparql11-overview/.

- “Spring/Summer 2012 (v.24 No.2/3) - National Information Standards Organization.” Accessed August 6, 2013. http://www.niso.org/publications/isq/2012/v24no2-3/.

- Summers, Ed, and Dorothea Salo. Linking Things on the Web: A Pragmatic Examination of Linked Data for Libraries, Archives and Museums. ArXiv e-print, February 19, 2013. http://arxiv.org/abs/1302.4591.

- “The Linking Open Data Cloud Diagram.” Accessed November 3, 2013. http://lod-cloud.net/.

- “The Trouble with Triples | Duke Collaboratory for Classics Computing (DC3).” Accessed November 6, 2013. http://blogs.library.duke.edu/dcthree/2013/07/27/the-trouble-with-triples/.

- Tim Berners-Lee, James Hendler, and Ora Lassila. “The Semantic Web.” Accessed September 4, 2013. http://www.scientificamerican.com/article.cfm?id=the-semantic-web.

- “Transforming EAD XML into RDF/XML Using XSLT.” Accessed August 16, 2013. http://archiveshub.ac.uk/locah/tag/transform/.

- “Triplestore - Wikipedia, the Free Encyclopedia.” Accessed November 11, 2013. http://en.wikipedia.org/wiki/Triplestore.

- “Turtle (syntax).” Wikipedia, the Free Encyclopedia, July 13, 2013. http://en.wikipedia.org/w/index.php?title=Turtle_(syntax)&oldid=542183836.

- Volz, Julius, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. “Discovering and Maintaining Links on the Web of Data.” In The Semantic Web - ISWC 2009, edited by Abraham Bernstein, David R. Karger, Tom Heath, Lee Feigenbaum, Diana Maynard, Enrico Motta, and Krishnaprasad Thirunarayan, 5823:650–665. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. http://www.springerlink.com/index/10.1007/978-3-642-04930-9_41.

- Voss, Jon. “LODLAM State of Affairs.” Information Standards Quarterly 24, no. 2/3 (2012): 41. doi:10.3789/isqv24n2-3.2012.07.

- W3C. “LinkedData.” Accessed August 4, 2013. http://www.w3.org/wiki/LinkedData.

- “Welcome to Euclid.” Accessed September 4, 2013. http://www.euclid-project.eu/.

- “Wiki.dbpedia.org : About.” Accessed November 3, 2013. http://dbpedia.org/About.

Appendix A: Code (ead2rdf.pl)

```
#!/usr/bin/perl

# ead2rdf.pl - make EAD files accessible via linked data

# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 6, 2013 - based on marc2linkedata.pl


# configure
use constant ROOT     => '/disk01/www/html/main/sandbox/liam';
use constant EAD      => ROOT . '/src/ead/';
use constant DATA     => ROOT . '/data/';
use constant PAGES    => ROOT . '/pages/';
use constant EAD2HTML => ROOT . '/etc/ead2html.xsl';
use constant EAD2RDF  => ROOT . '/etc/ead2rdf.xsl';
use constant SAXON    => 'java -jar /disk01/www/html/main/sandbox/liam/bin/saxon.jar -
s:##SOURCE## -xsl:##XSL## -o:##OUTPUT##';

# require
use strict;
use XML::XPath;
use XML::LibXML;
use XML::LibXSLT;

# initialize
my $saxon  = '';
my $xsl    = '';
my $parser = XML::LibXML->new;
my $xslt   = XML::LibXSLT->new;

# process each record in the EAD directory
my @files = glob EAD . "*.xml";
for ( 0 .. $#files ) {

  # re-initialize
  my $ead = $files[ $_ ];
  print "         EAD: $ead\n";

  # get the identifier
```

```
my $xpath      = XML::XPath->new( filename => $ead );
my $identifier = $xpath->findvalue( '/ead/eadheader/eadid' );
$identifier    =~ s/[^\w ]//g;
print "  identifier: $identifier\n";
print "         URI: http://infomotions.com/sandbox/liam/id/$identifier\n";

# re-initialize and sanity check
my $output = PAGES . "$identifier.html";
if ( ! -e $output or -s $output == 0 ) {

  # transform marcxml into html
  print "        HTML: $output\n";
  my $source     = $parser->parse_file( $ead )  or warn $!;
  my $style      = $parser->parse_file( EAD2HTML )   or warn $!;
  my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
  my $results    = $stylesheet->transform( $source )  or warn $!;
  my $html       = $stylesheet->output_string( $results );

  &save( $output, $html );

}
else { print "        HTML: skipping\n" }

# re-initialize and sanity check
my $output = DATA . "$identifier.rdf";
if ( ! -e $output or -s $output == 0 ) {

  # create saxon command, and save rdf
  print "         RDF: $output\n";
  $saxon  =  SAXON;
  $xsl    =  EAD2RDF;
  $saxon  =~ s/##SOURCE##/$ead/e;
  $saxon  =~ s/##XSL##/$xsl/e;
  $saxon  =~ s/##OUTPUT##/$output/e;
  system $saxon;

}
else { print "         RDF: skipping\n" }

# prettify
print "\n";
```

```
}

# done
exit;


sub save {

  open F, ' > ' . shift or die $!;
  binmode( F, ':utf8' );
  print F shift;
  close F;
  return;

}
```

Appendix: A question from a library school student

In January of 2014 I received the following email message from a library school student. The questions they asked were apropos to the Guide, so I thought I'd include my responseggt here, but the names have been changed to protect the innocent.

```
From: Eric Lease Morgan <emorgan@nd.edu>
Subject: Re: RDF ontologies discussion on Code4Lib Listserv
Date: January 21, 2014 at 9:36:36 PM EST

> I'm writing you to ask you about your thoughts on implementing
> these kinds of RDF descriptions for institutional collections.
> Have you worked on a project that incorporated LD technologies
> like these descriptions? What was that experience like? Also,
> what kind of strategies have you used to implement these
> strategies, for instance, was considerable buy-in from your
> organization necessary, or were you able to spearhead it
> relatively solo? In essence, what would it "cost" to really do
> this?
>
> I apologize for the mass of questions, especially over e-mail. My
> only experience with ontology work has been theoretical, and I
> haven't met any professionals in the field yet who have actually
> used it. When I talk to my mentors about it, they are aware of it
> from an academic standpoint but are wary of it due these
> questions of cost and resource allocation, or confusion that it
> doesn't provide anything new for users. My final project was to
> build an ontology to describe some sort of resource and I settled
> on building a vocabulary to describe digital facsimiles and their
> physical artifacts, but I have yet to actually implement or use
> any of it. Nor have I had a chance yet to really use any
> preexisting vocabularies. So I've found myself in a slightly
> frustrating position where I've studied this from an academic
> perspective and seek to incorporate it in my GLAM work, but I
> lack the hands-on opportunity to play around with it.
>
> --
> MLIS Candidate

Dear MLS Candidate, no problem, really, but I don't know how much help I will really
be.

The whole RDF / Semantic Web thing started more than ten years ago. The idea was to
expose RDF/XML, allow robots to crawl these files, amass the data, and discover new
knowledge – relationships – underneath. Many in the library profession thought this
```

was science fiction and/or the sure pathway to professional obsolescence. Needless to say, it didn’t get very far. A few years ago the idea of linked data was articulated, and it a nutshell it outlined how to make various flavors of serialized RDF available via an HTTP technique called content negotiation. This was when things like Turtle, N3, the idea of triple stores, maybe SPARQL, and other things came to fruition. This time the idea of linked data was more real and got a bit more traction, but it is still not main stream.

I have very little experience putting the idea of RDF and linked data into practice. A long time ago I created RDF versions of my Alex Catalogue and implemented a content negotiation tool against it. The Catalogue was not a part of any institution other than myself. When I saw the call for the LiAM Guidebook I applied and got the “job” because of my Alex Catalogue experiences as well as some experience with a thing colloquially called The Catholic Portal which contains content from EAD files.

I knew this previously, but linked data is all about URIs and ontologies. Minting URIs is not difficult, but instead of rolling your own, it is better to use the URIs of others, such as the URIs in DBpedia, GeoNames, VIAF, etc. The ontologies used to create relationships between the URIs are difficult to identify, articulate, and/or use consistently. They are manifestations of human language, and human language is ambiguous. Trying to implement the nuances of human language in computer “sentences” called RDF triples is only an approximation at best. I sometimes wonder if the whole thing can really come to fruition. I look at OAI-PMH. It had the same goals, but it was finally called not a success because it was too difficult to implement. The Semantic Web may follow suit.

That said, it is not too difficult to make the metadata of just about any library or archive available as linked data. The technology is inexpensive and already there. The implementation will not necessarily implement best practices, but it will not expose incorrect nor invalid data, just data that is not the best. Assuming the library has MARC or EAD files, it is possible to use XSL to transform the metadata into RDF/XML. HTML and RDF/XML versions of the metadata can then be saved on an HTTP file system and disseminated either to humans or robots through content negotiation. Once a library or archive gets this far they can then either improve their MARC or EAD files to include more URIs, they can improve their XSLT to take better advantage of shared ontologies, and/or they can dump MARC and EAD all together and learn to expose linked data directly from (relational) databases. It is an iterative process which is never completed.

Nothing new to users? Ah, that is the rub and a sticking point with the linked data / Semantic Web thing. It is a sort of chicken & egg problem. “Show me the cool application that can be created if I expose my metadata as linked data”, say some people. On the other hand, “I can not create the cool application until there is a critical mass of available content.” Despite this issue, things are happening for readers, namely mash-ups. (I don’t like the word “users”.) Do a search in Facebook for the Athens. You will get a cool looking Web page describing Athens, who has been there, etc. This was created by assembling metadata from a host of different places (all puns intended), and one of those places were linked data repositories. Do a search in Google for the same thing. Instead of just bringing back a list of links, Google presents you with real content, again, amassed through various APIs including linked data. Visit VIAF and search for a well-known author. Navigate the result an you will maybe end up at WorldCat identities where all sorts of interesting information about an author, who they wrote with, what they wrote, and where is displayed. All of this is rooted in linked data and Web Services computing techniques. This is where the benefit comes. Library and archival metadata will become part of these mash-up – called “named graphs” – driving readers to library and archival websites. Linked data can become part of Wikipedia. It can be used to enrich descriptions of people in authority lists, gazetteers, etc.

What is the cost? Good question. Time is the biggest expense. If a person knows what they are doing, then a robust set of linked data could be exposed in less than a month, but in order to get that far many people need to contribute. Systems types to get the data out of content management systems as well as set up HTTP servers. Programmers will be needed to do the transformations. Catalogers will be needed to assist in the interpretation of AACR2 cataloging practices, etc. It will take a village to do the work, and that doesn’t even count convincing people this is a good idea.

Your frustration is not uncommon. Often times if there is not a real world problem to solve, learning anything new when it comes to computers is difficult. I took BASIC computer programming three times before anything sunk in, and it only sunk in when I needed to calculate how much money I was earning as a taxi driver.

Try implementing one of your passions. Do you collect anything? Baseball cards? Flowers? Books? Records? Music? Art? Is there something in your employer’s special collections of interest to you? Find something of interest to you. For simplicity’s sake, use a database to describe each item in the collection making sure each record in our database includes a unique key field. Identify one or more ontologies (others as well as rolling your own) whose properties match closely the names of your fields in your database. Write a program against your database to create static HTML pages. Put the pages on the Web. Write a program against your database to create static RDF/XML (or some other RDF serialization). Put the pages on the Web. Implement a content negotiation script that takes the keys to your database’s fields as input and redirects HTTP user agents to either the HTML or RDF. Submit the root of your linked data implementation to Datahub.io. Ta da! You have successfully implemented linked data and learned a whole lot along the way. Once you get that far you can take what you have learned and apply it in a bigger and better way for a larger set of data.

On one hand the process is not difficult. It is a matter of repurposing the already existing skills of people who work in cultural heritage institutions. On the other hand, change in the ways things are done is difficult (but not as difficult in the what of what is done). The change is difficult to balance existing priorities.

Exposing library and archival content as linked data represents a different working style, but the end result is the same – making the content of our collections available for use and understanding.

HTH.

–
Eric Morgan

Appendix: cURL and content-negotiation

This is the tiniest introduction to cURL and content-negotiation. CURL is a command-line tool making it easier for you to see the Web as data and not presentation. It is a sort of Web browser, but more specifically, it is a thing called a user-agent. Content-negotiation is an essential technique for publishing and making accessible linked data. Please don't be afraid of the command-line though. Understanding how to use cURL and do content-negotiation by hand will take you a long way in understanding linked data.

The first step is to download and install cURL. If you have a Macintosh or a Linux computer, then it is probably already installed. If not, then give the cURL download wizard a whirl.[1] We'll wait.

Next, you need to open a terminal. On Macintosh computers a terminal application is located in the Utilities folder of your Applications folder. It is called "Terminal". People using Windows-based computers can find the "Command" application by searching for it in the Start Menu. Once cURL has been installed and a terminal has been opened, then you can type the following command at the prompt to display a help text:

```
curl --help
```

There are many options there, almost too many. It is often useful to view only one page of text at a time, and you can "pipe" the output through to a program called "more" to do this. By pressing the space bar, you can go forward in the display. By pressing "b" you can go backwards, and by pressing "q" you can quit:

```
curl --help | more
```

Feed cURL the complete URL of Google's home page to see how much content actually goes into their "simple" presentation:

```
curl http://www.google.com/ | more
```

[1] The cURL download wizard is available at http://curl.haxx.se/dlwiz/.

The communication of the World Wide Web (the hypertext transfer protocol or HTTP) is divided into two parts: 1) a header, and 2) a body. By default cURL displays the body content. To see the header, add the -I (for a mnemonic, think “information”) to the command:

```
  curl -I http://www.google.com/
```

The result will be a list of characteristics the remote Web server is using to describe this particular interaction between itself and you. The most important things to note are: 1) the status line and 2) the content type. The status line will be the first line in the result, and it will say something like “HTTP/1.1 200 OK”, meaning there were no errors. Another line will begin with “Content-Type:” and denotes the format of the data being transferred. In most cases the content type line will include something like “text/html” meaning the content being sent is in the form of an HTML document.

Now feed cURL a URI for Walt Disney, such as one from DBpedia:

```
  curl http://dbpedia.org/resource/Walt_Disney
```

The result will be empty, but upon the use of the -I switch you can see how the status line changed to “HTTP/1.1 303 See Other”. This means there is no content at the given URI, and the line starting with “Location:” is a pointer — an instruction — to go to a different document. In the parlance of HTTP this is called redirection. Using cURL going to the recommended location results in a stream of HTML:

```
  curl http://dbpedia.org/page/Walt_Disney | more
```

Most Web browsers automatically follow HTTP redirection commands, but cURL needs to be told this explicitly through the use of the -L switch. (Think “location”.) Consequently, given the original URI, the following command will display HTML even though the URI has no content:

```
  curl -L http://dbpedia.org/resource/Walt_Disney | more
```

Now remember, the Semantic Web and linked data depend on the exchange of RDF, and upon closer examination you can see there are “link” elements in the resulting

HTML, and these elements point to URLs with the .rdf extension. Feed these URLs to cURL to see an RDF representation of the Walt Disney data:

```
  curl http://dbpedia.org/data/Walt_Disney.rdf | more
```

Downloading entire HTML streams, parsing them for link elements containing URLs of RDF, and then requesting the RDF is not nearly as efficient as requesting RDF from the remote server in the first place. This can be done by telling the remote server you accept RDF as a format type. This is accomplished through the use of the -H switch. (Think "header".) For example, feed cURL the URI for Walt Disney and specify your desire for RDF/XML:

```
  curl -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Walt_Disney
```

Ironically, the result will be empty, and upon examination of the HTTP headers (remember the -I switch) you can see that the RDF is located at a different URL, namely, http://dbpedia.org/data/Walt_Disney.xml:

```
  curl -I -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Walt_Disney
```

Finally, using the -L switch, you can use the URI for Walt Disney to request the RDF directly:

```
  curl -L -H "Accept: application/rdf+xml" http://dbpedia.org/resource/Walt_Disney
```

That is cURL and content-negotiation in a nutshell. A user-agent submits a URI to a remote HTTP server and specifies the type of content it desires. The HTTP server responds with URLs denoting the location of desired content. The user-agent then makes a more specific request. It is sort of like the movie. "One URI to rule them all." In summary, remember:

- cURL is a command-line user-agent
- given a URL, cURL returns, by default, the body of an HTTP transaction
- the -I switch allows you to see the HTTP header
- the -L switch makes cURL automatically follow HTTP redirection requests

- the -H switch allows you to specify the type of content you wish to accept
- given a URI and the use of the -L and -H switches you are able to retrieve either HTML or RDF

Use cURL to actually see linked data in action, and here are a few more URIs to explore:

- Walt Disney via VIAF - http://viaf.org/viaf/36927108/
- origami via the Library of Congress - http://id.loc.gov/authorities/subjects/sh85095643
- Paris from DBpedia - http://dbpedia.org/resource/Paris

Appendix: “Linked Open Data”

For all intents and contexts, linked data and linked open data are synonymous, but the subtle difference does need to be discussed. Linked open data is a qualification of linked data. Linked open data comes with an explicit license agreement denoting how the accessible data can be “freely” used. In this case, the words “free” and “open” are analogous to free “open source software”. Just as open source software is available for use and re-use, linked open data is free for use and re-use. Attribution needs to be made. The data can be freely used. While copyrights may still be in place when it comes to linked open data, the copyrights allow for the use, re-use, and re-distribution. The intent of linked open data is to use the content in ways that it canb e used in many ways for many purposes. While the distinction between linked data and linked open data are may be large in the eyes of some people, for simplicities sake, the phrase linked data is synonymous with linked open data, even though some feel the distinction needs to be delineated to a greater degree.

Appendix: Content-negotiation, REST-ful style computing, and APIs

It is important to understand the difference between linked data and more proprietary application programmer interfaces. This difference is really between REST-ful interfaces to data and content negotiaion interfaces. REST-ful interfaces are specific to specific websites. Content-negotiation is more universal and does not require API keys or the knowledge of name-value pairs. Linked data computing techniques is our HTTP. REST-ful interfaces need to be learn and are all different from one another. Linked data is more universal.

Appendix: Benefits

From: Ingrid Mason <ingrid.b.mason@gmail.com>
Subject: Re: [LODLAM] quick benefits to hosting instutitions
Date: January 22, 2014 at 9:49:47 PM EST
To: lod-lam@googlegroups.com
Reply-To: lod-lam@googlegroups.com

Hi Jody,

If I understand correctly, you're keen to find examples of value generated. I'll give this a whirl... and see if I'm being helpful.

Converting data that already exists, >> providing a data service that a user community seeks to reuse and contribute to (access value).

PeopleAustralia (National Library of Australia) provides permanent, resolvable unique identifiers that link to records about parties, i.e. people or organisations. The authority file at the Library already had, was reused. This data source was enhanced as a result of ANDS funding to identify parties that manage or own research data collections. You'll see that this has increased the capacity for discovery (through collaboration with other data providers). Key contact: Tim Sherratt @wragge

Collaborating with custodians of primary material (collection managers), using third party data (Dbpedia), and finding, identifying and linking entities in the data >> brought to light information that was previously unknown (research value).

LinkedJazz at the Pratt Institute. Finalist in the LODLAM 2013 summit award along with some other folks (for being generally super clever with LOD things). Key contact: Cristina Patuelli @cristinapattuel

Providing insights into the links in your own data >> improve data quality (data value).

Check out Chris McDowall's post on linking data in digitalNZ. @fogonwater

There are other kinds of value in all that. Guess benefits depend on what the strategic goals of the organisation are, and what the research community needs in terms of access.

Hope that helps?

Ingrid

On 23 January 2014 12:55, Jody DeRidder <jody@jodyderidder.com> wrote:
Hi --
I'm looking for selling points for utilizing linked data, for the institutions creating it. I would appreciate being pointed towards demos that show how particular tools can be utilized to provide improved access and use of local content for which linked data exists -- particularly with regards to primary source materials which have been digitized.
Without the "benefits" side of the equation, it's difficult to make a case for the "costs" part of the work, fascinating as it may be.
Suggestions?   Please feel free to contact me off list.
Thanks!
Jody L. DeRidder
Head, Digital Services
University of Alabama Libraries
Tuscaloosa, AL 35487
Phone: 205.348.0511

"Hope lies in dreams, in imagination, and in the courage of those who dare to make dreams into reality."
--Jonas Salk

Appendix: A Thing or two EAD and linked data

From Ben Companjen <ben.companjen@dans.knaw.nl>

While I'm no archivist by training (information systems engineer I am), I've learned a thing or two from having to work with EAD and its basis for use, ISAD(G) (all citations below are from ISAD(G), 2nd edition). As with all information modelling, either inside or outside the Linked Data domain, you should take a step back to look at the goal of the description. When you have a list of what you want to describe, you can start looking for ontologies.

You probably know this, but I was triggered by "Because many archival descriptions are rooted in MARC records, and MODS is easily mapped from MARC." to respond. IMO archival descriptions are rooted in rules for description, not a specific file format.

So, when I [see] of (some of) the essences of archival description, I think of:

* "The purpose of archival description is to identify and explain the context and content of archival material in order to promote its accessibility. This is achieved by creating accurate and appropriate representations and by organizing them in accordance with predetermined models." (§I.2)

* "… seven areas of descriptive information:

1. Identity Statement Area (where essential information is conveyed to identify the unit of description)

2. Context Area (where information is conveyed about the origin and custody of the unit of description)

3. Content and Structure Area (where information is conveyed about the subject matter and arrangement of the unit of description)

4. Condition of Access and Use Area (where information is conveyed about the availability of the unit of description)

5. Allied Materials Area (where information is conveyed about materials having an important relationship to the unit of description)

6. Note Area (where specialized information and information that cannot be accommodated in any of the other areas may be conveyed).

7. Description Control Area (where information is conveyed on how, when and by whom the archival description was prepared)." (§I.11)

There is a distinction between the thing being described, and the description itself, and both have an important role within the archival description. (If anything so far causes confusion with anyone here, I misunderstood and accept to be corrected :)) NB: this is one way of thinking of descriptions. Incorporating the PROV-ontology would make sense for expressing more/other aspects of the provenance of archival entities, but I haven't got round to becoming an expert of PROV yet ;)

ISAD(G) lists 26 "elements that may be combined to constitute the description of an archival entity".

Trying to translate these 'elements', I'd end up with possible a lot more than 26 RDFS/OWL properties.

Depending on the type of archival entity you can/should of course use more specific ontologies.

Let me list some properties and related ontologies.

* Identity statement area

o Identifiers - The URI, naturally, and other IDs. Could be linked using dc(terms):identifier, or mods:identifier, or other ontologies. Ideally there is some way of linking the domain of the ID to the ID itself, because "box 101" is likely not unique in the universe. Perhaps you want to publish a URI strategy separately to explain how the URI was assembled/derived.

o Title - Again DC(terms), MODS, RDA

o Date(s) - You want properties that have a clear meaning. For example, dcterms:created and mods:dateCreated assume it is clear what "when the resource was created" means. DC terms are vague, I mean general, on purpose. You could create some properties `owl:subPropertyOf` dcterms date properties for this. I'd look into EDTF for encoding uncertain dates and ranges and BCE dates (MODS doesn't support BCE dates).

o Level of description - What kind of 'documentary unit' does the description describe? A whole building's content or one piece of paper? I don't know of any ontology with terms "fonds", …, "file", "item", but you could say `<http URI> rdf:type <fonds class URI>`.

o Extent and medium - Saying anything about extent and medium should possible only happen on the lowest level of description. Any higher level extent and medium should be calculated by aggregating lower level descriptions. On the lowest level, refer to class URIs. A combination of dimensions and material {c|sh}ould be a class, e.g. A4 paper 80 grams/square meter.

* Context area

o Creator(s) and administrative/biographical history - As ISAD(G) refers to ISAAR(CPF) for description of corporate bodies, people, and families, this is a perfect example of using existing people- and organisation-describing ontologies like FOAF, BIO, ORG, and others are useful for separate descriptions of the people and organisations involved. You want specific properties to describe the roles of these 'agents' in the history of the archival entity…

o Archival history and Immediate source of acquisition or transfer - … and you would want them 'here' (of course there is no particular order in which these properties are used). PREMIS and PROV come to mind first for recording who did what to what, (where and?) when and with what result. There are probably some ontologies describing possible "events" as RDFS/OWL classes, so you could link to those. The immediate source of acquisition or transfer may be just another event.

* Content and structure area

o Scope and content - Descriptions, keywords, terms from authority files about "scope (such as, time periods, geography) and content, (such as documentary forms, subject matter, administrative processes) … appropriate to the level of description.": pretty natural fit for links to SKOS thesauri and other ontologies of real-world 'things'. One might think of dcterms:subject, dcterms:description, dcterms:temporalCoverage etc., but describing *how* exactly such terms relate to the archival entity needs more specific properties than "subject" et al.

o Appraisal, destruction and scheduling information - Reasons for including things and (possibly) removal of archival entities should go very well in rules, and some types of rules go very well in ontologies. Making this up as I type: <class of letters written by the head of state> rdfs:subClassOf <class of 'things to be kept'>. The actual selection and destruction actions could be modelled in the same way as other actions are described for provenance.

o Accruals - Whether more content can be expected probably depends on other properties of the archival entity, like its type(s) and creator(s). I don't know about specific properties to record this, but <class of living heads of state archival entities> rdfs:subClassOf <class of 'living' archival entities>? There are ways of modelling rules for this, like the Rules Interchange Format, but the rules may be defined by the archives and archivists.

o System of arrangement - Thinking about this, I tend to think of a collection of keywords to describe the arrangement of a low-level archival entity like a folder or box: alphabetical, as found on deceased's desk. But there is more, of course. Perhaps using the Collection Ontology for low levels could help generate higher level 'systems of arrangement'.

* Conditions of access and use area

o Conditions governing access and Conditions governing reproduction - You can describe rights with the Creative Commons Rights Expression Language.

o Language of material - mods:language maybe? Preferably used on sub-document level and generated for higher-level descriptions.

o Physical characteristics / technical requirements - Conditions should follow from their respective properties: <class of PDF/A-1b files> ..:requiresForReading <class of PDF/A-1b readers> and rules that say documents in <class A> are embargoed for 20 years after creation + a creation date can present enough information to the agent to determine dcterms:dateAvailable.

o Finding aids - As a non-archivist I had some trouble understanding the difference between descriptions and finding aids and what the exact use of a finding aid was. Also, having grown up with search engines, indexes, I think the concept may eventually become extinct. I guess you could use foaf:page to link a document-like finding aid to the archival entity and rdfs:seeAlso to point to machine-actionable related things.

* Allied materials area

o Existence and location of originals/copies - PROV can be used to link a copy to an original (and how the copy was created etc.). `<X> prov:wasDerivedFrom <Y>.
<Y> :isAt <AnotherArchive>.`

o Related units of description / Publication note - Use properties that describe the specific relations among archival entities. DC Terms has some useful ones, like for citations. Related items can be derived from all or selected properties automatically too.

* Notes area

o Notes - dcterms:description? Unlike a document containing rules that needs to be finished at some time, Linked Data has no such rule. You can always create a property with a well-defined meaning to use for specific information.

* Description control area

o Archivist's note / dates of description - Who did what when, where, why and how to the description itself. Same as for the unit of description itself. This may be a good time to draw a bit more attention to the question: *what is a description?* I don't have a (/ there is no) final answer, but as The One True Written Paper Description from long ago is becoming a set of triples, you want to think about it. You could link versions of RDF documents using PROV to record this information.

o Rules and conventions - A link to the rules and conventions for description. Could also fit with the PROV provenance.

No, this is not a list of ontologies to use/explore right away, but I hope you (and others) find it helpful, or perhaps even food for discussion. Also, have a look at CIDOC-CRM. It has lots of properties.

[1] ISAD(G) (http://bit.ly/1mmXMmJ) - "This standard provides general guidance for the preparation of archival descriptions. It is to be used in conjunction with existing national standards or as the basis for the development of national standards."

[2] IDOC-CRM (http://www.cidoc-crm.org) - "The CIDOC Conceptual Reference Model (CRM) provides definitions and a formal structure for describing the implicit and explicit concepts and relationships used in cultural heritage documentation.

-- Ben

Appendix: RDA and linked data

From: Jon Phipps <jphipps@MADCREEK.COM>
Subject: Re: [CODE4LIB] Fwd: [rules] Publication of the RDA Element Vocabularies
Date: January 22, 2014 at 6:26:19 PM EST
To: "CODE4LIB@LISTSERV.ND.EDU" <CODE4LIB@LISTSERV.ND.EDU>
Reply-To: Code for Libraries <CODE4LIB@LISTSERV.ND.EDU>

Hi Dan,

Thanks for taking such an interest!

Regarding your questions and concerns:

'slash' vs. 'hash' URIs:
As a matter of design, we coin URIs for retrieval of information about the resource identified by the URI by machines, not humans. The most current formal rules[1] state that retrieving a 'slash' fragment should return just that fragment when resolved. We're currently breaking that rule by always returning the entire vocabulary, as if it was indeed using hash URIs and will fix it in the next few weeks. An example of such a fragment (generated by the Open Metadata Registry for
http://rdaregistry.info/Elements/w/P10001)
is here:
http://metadataregistry.org/schemaprop/show/id/15304.rdf

We believe, as a matter of good design, that URIs coined for large vocabularies should minimize retrieval bandwidth, particularly since it's highly unlikely that the entire vocabulary will (or should) be retrieved when the properties are used individually as part of an application profile. The entire vocabulary can always be acquired by requesting it from the vocabulary's namespace URI:
http://rdaregistry.info/Elements/w/

Lexical (readable, but not semantic) URIs:

One of the most common misuses of vocabularies is the misunderstanding of
the semantics of the property identified by the URI based on the user's
personal, colloquial, or domain-specific interpretation of the semantics of
the URI (dc:title is the one I've seem misused most often). So we believe
that good vocabulary design _should_ obscure the semantics requiring that
the actual vocabulary documentation be viewed by a human.

The other problem is that the 'semantics' are most often broadly identified
with the lexical label used in the URI. Vocabularies, no matter how stable
semantically, _will_ evolve and that evolution often results in a change to
the label(s), even if the semantics communicated by the URI don't change.

And then there's the issue of spelling (British English vs. American
English) and language. Should we assume that the entire world must use, and
understand English in order to effectively use a vocabulary? We don't
think so.

To at least partially address this we have coined multiple URIs for each
property, as explained here:
http://www.rdaregistry.info/Elements/e/
"All RDA URIs have both an immutable canonical form and a 'readable',
lexical form, which is subject to change (changes will be redirected)." The
lexical URIs follow the naming convention you identified and are largely
based on the current English (British) label.

Content-type: application/octet-stream:
We just got the server (nginx) setup yesterday and we haven't yet set the
mime types correctly. Again we'll fix that very shortly.

Jon Phipps
Metadata Management Associates
Open Metadata Registry

[1] http://www.w3.org/TR/swbp-vocab-pub/

Jon

---------- Forwarded message ----------
From: JSC Secretary <jscsecretary@rdatoolkit.org>
Date: Tue, Jan 21, 2014 at 10:23 AM
Subject: [rules] Publication of the RDA Element Vocabularies
<snip recipients>

RDA colleagues,

See the announcement below, also posted on the JSC website. Feel free to share this information with your colleagues.

Regards, Judy Kuhagen

= = = = =

The Joint Steering Committee for Development of RDA (JSC), Metadata Management Associates, and ALA Publishing (on behalf of the co-publishers of RDA) are pleased to announce that the RDA elements and relationship designators have been published in the Open Metadata Registry (OMR) as Resource Description Framework (RDF) element sets suitable for linked data
and semantic Web applications.

The elements include versions "unconstrained" by Functional Requirements for Bibliographic Records (FRBR) and Functional Requirements for Authority
Data (FRAD), the standard library models underpinning RDA, that are intended for use in applications by non-RDA communities.

The published version of the RDA element sets builds on several years of work by the DCMI/RDA Task Group. Earlier versions developed by the Group

will remain available, but will be deprecated for further development and
use, and redirected to the new version.

Gordon Dunsire, Chair of the JSC, said "The RDA element set is a
distillation of modern approaches to resource discovery supporting rich
descriptions of library and cultural heritage materials and detailed
relationships between them at international level. The JSC has recently
established a working group to assist in extending and refining the RDA
elements, and hopes that they will be useful to other communities,
ranging
from close neighbours in library linked data to the global networks of
general search."

Diane Hillmann of Metadata Management Associates said "We are extremely
pleased to be able to make this new version available now in fully
published form, ready for implementation by libraries and vendors. We
look
forward to discussing the important features available in this version
with
our colleagues at the upcoming ALA Midwinter meetings and beyond."

James Hennelly, Managing Editor of RDA Toolkit, said "This is an
important
update to the RDA Registry and a crucial step in the advancement of RDA's
mission to be a standard that is accessible to both cataloging
professionals, through the toolkit and print and ebook publications, and
to
application developers seeking to make use of library data, through the
Registry's expression of the RDA elements and vocabularies."

The basic RDA element set namespace is rdaregistry.info and it contains
a
total of over 1600 properties and classes. Elements are distributed in
sets
(the number of elements in each set is given in brackets):
Agent properties

[http://metadataregistry.org/schema/show/id/81.html]<
http://metadataregistry.org/schema/show/id/81.html>(226)
Expression properties
[http://metadataregistry.org/schema/show/id/78.html]<
http://metadataregistry.org/schema/show/id/78.html>(236)
Item properties
[http://metadataregistry.org/schema/show/id/80.html]<
http://metadataregistry.org/schema/show/id/80.html>(54)
Manifestation properties
[http://metadataregistry.org/schema/show/id/79.html]<
http://metadataregistry.org/schema/show/id/79.html>(213)
Work properties
[http://metadataregistry.org/schema/show/id/77.html]<
http://metadataregistry.org/schema/show/id/77.html>(232)
Unconstrained properties
[http://metadataregistry.org/schema/show/id/82.html]<
http://metadataregistry.org/schema/show/id/82.html>(698)
Classes [http://metadataregistry.org/schema/show/id/83.html]<
http://metadataregistry.org/schema/show/id/83.html>(8)

Follow the links to see details of each element set.

Questions or comments on the content of the element sets may be addressed to the Chair of the JSC, Gordon Dunsire [jschair@rdatoolkit.org].
Questions
and comments on the encoding of the vocabularies or on the Open Metadata Registry may be addressed to Diane Hillmann [metadata.maven@gmail.com].

Appendix: Archival materials and linked data

Links describing the why's and wherefore's of LOHAC's vocabulary:

- http://archiveshub.ac.uk/locah/2011/03/describing-the-things-the-rdf-terms-used-part-1/
- http://archiveshub.ac.uk/locah/2011/03/describing-the-things-the-rdf-terms-used-part-2/

Appendix: RDF serializations

RDF can be expressed in many different formats, called “serializations”.

RDF (Resource Description Framework) is a conceptual data model made up of “sentences” called triples — subjects, predicates, and objects. Subjects are expected to be URIs. Objects are expected to be URIs or string literals (think words, phrases, or numbers). Predicates are “verbs” establishing relationships between the subjects and the objects. Each triple is intended to denote a specific fact.

When the idea of the Semantic Web was first articulated XML was the predominant data structure of the time. It was seen as a way to encapsulate data that was both readable by humans as well as computers. Like any data structure, XML has both its advantages as well as disadvantages. On one hand it is easy to determine whether or not XML files are well-formed, meaning they are syntactically correct. Given a DTD, or better yet, an XML schema, it is also easy determine whether or not an XML file is valid — meaning does it contain the necessary XML elements, attributes, and are they arranged and used in the agreed upon manner. XML also lends itself to transformations into other plain text documents through the generic, platform-independent XSLT (Extensible Stylesheet Language Transformation) process. Consequently, RDF was originally manifested — made real and “serialized” — though the use of RDF/XML.

The example of RDF at the beginning of the Guidebook was an RDF/XML serialization:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dcterms="http://purl.org/dc/terms/"
         xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <rdf:Description rdf:about="http://en.wikipedia.org/wiki/
Declaration_of_Independence">
    <dcterms:creator>
      <foaf:Person rdf:about="http://id.loc.gov/authorities/names/n79089957">
        <foaf:gender>male</foaf:gender>
      </foaf:Person>
    </dcterms:creator>
  </rdf:Description>
</rdf:RDF>
```

On the other hand, XML, almost by definition, is verbose. Element names are expected to be human-readable and meaningful, not obtuse nor opaque. The judicious use of special characters (&, <, >, “, and ‘) as well as entities only adds to the difficulty of actually reading XML. Consequently, almost from the very beginning people thought RDF/XML was not the best way to express RDF, and since then a number of other syntaxes — data structures — have manifested themselves.

Below is the same RDF serialized in a format called Notation 3 (N3), which is very human readable, but not extraordinarily structured enough for computer processing. It incorporates the use of a line-based data structure called N-Triples used to denote the triples themselves:

```
@prefix foaf: <http://xmlns.com/foaf/0.1/>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dcterms: <http://purl.org/dc/terms/>.
<http://en.wikipedia.org/wiki/Declaration_of_Independence> dcterms:creator <http://
id.loc.gov/authorities/names/n79089957>.
<http://id.loc.gov/authorities/names/n79089957> a foaf:Person;
       foaf:gender "male".
```

JSON (JavaScript Object Notation) is a popular data structure inherent to the use of JavaScript and Web browsers, and RDF can be expressed in a JSON format as well:

```
{
  "http://en.wikipedia.org/wiki/Declaration_of_Independence": {
    "http://purl.org/dc/terms/creator": [
      {
        "type": "uri",
        "value": "http://id.loc.gov/authorities/names/n79089957"
      }
    ]
  },
  "http://id.loc.gov/authorities/names/n79089957": {
    "http://xmlns.com/foaf/0.1/gender": [
      {
        "type": "literal",
        "value": "male"
      }
    ],
    "http://www.w3.org/1999/02/22-rdf-syntax-ns#type": [
      {
        "type": "uri",
        "value": "http://xmlns.com/foaf/0.1/Person"
      }
    ]
  }
}
```

Just about the newest RDF serialization is an embellishment of JSON called JSON-LD. Compare & contrasts the serialization below to the one above:

```
{
  "@graph": [
    {
      "@id": "http://en.wikipedia.org/wiki/Declaration_of_Independence",
      "http://purl.org/dc/terms/creator": {
        "@id": "http://id.loc.gov/authorities/names/n79089957"
      }
    },
    {
      "@id": "http://id.loc.gov/authorities/names/n79089957",
      "@type": "http://xmlns.com/foaf/0.1/Person",
      "http://xmlns.com/foaf/0.1/gender": "male"
    }
  ]
}
```

RDFa represents a way of expressing RDF embedded in HTML, and here is such an expression:

```
<div xmlns="http://www.w3.org/1999/xhtml"
  prefix="
    foaf: http://xmlns.com/foaf/0.1/
    rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
    dcterms: http://purl.org/dc/terms/
    rdfs: http://www.w3.org/2000/01/rdf-schema#"
  >
  <div typeof="rdfs:Resource" about="http://en.wikipedia.org/wiki/
Declaration_of_Independence">
    <div rel="dcterms:creator">
      <div typeof="foaf:Person" about="http://id.loc.gov/authorities/names/n79089957">
        <div property="foaf:gender" content="male"></div>
      </div>
    </div>
  </div>
</div>
```

The purpose of publishing linked data is to make RDF triples easily accessible. This does not necessarily mean the transformation of EAD or MARC into RDF/XML, but rather making accessible the statements of RDF within the context of the reader. In this case, the reader may be a human or some sort of computer program. Each serialization has its own strengths and weaknesses. Ideally the archive would figure out ways exploit each of these RDF serializations in specific publishing purposes.

For a good time, play with the RDF Translator which will convert one RDF serialization into another. [2]

The RDF serialization process also highlights how data structures are moving away from a document-centric models to a statement-central models. This too has consequences for way cultural heritage institutions, like archives, think about exposing their metadata, but that is the topic of another essay.

[2] The RDF Translator is authored by Alex Stolz and found at http://rdf-translator.appspot.com.

Appendix: Vocabularies and ontologies

Look for vocabularies at Linked Open Vocabularies (LOV) — http://lov.okfn.org/dataset/lov/

Consider also Aaron Rubinstiens “Archival collections ontology" — http://gslis.simmons.edu/archival/arch/index.html

or mayby The LOCAH RDF Vocabulary = http://data.archiveshub.ac.uk/def/
The building blocks of linked data include:

- URIs pointing to real-world objects: people, places, or things where things can be ideas or just about anything on the Web
- Ontologies, the language(s) of relationships between the URIs
- Content to share with the wider world
- People to do the work
- Computer technology to manifest the work

Probably one of the more difficult intellectual tasks you will have when it comes to making your content available as linked data will be the selection of one or more ontologies used to make relationship between the subjects and objects of your triples. Probably the easiest way to think about these ontologies is as if they were fields in a MARC record or EAD file. Such an analogy is useful, but not 100% correct. Probably the best way to think of the ontologies is as if they were verbs in a sentence denoting relationships between things — subjects and objects. But if ontologies are sets of “verbs”, then they are akin to human language, and human language is ambiguous. Therein lies the difficulty with ontologies. There is no “right” way to implement them. Instead, there is only best or common practice. There are no hard and fast rules. Everything comes with a bit of interpretation. The application and use of ontologies is very much like the application and use of written language in general. In order for written language to work well two equally important things need to happen. First, the writer needs to be able to write. They need to be able to choose the most appropriate language for their intended audience. Shakespeare is not “right” with his descriptions of love, but instead his descriptions of love (and many other human emotions) resinate with a very large number of people. Second, written language requires the reader to have a particular adeptness as well. Shakespeare can not be expected to write one thing and communicate to everybody. The reader needs to understand English, or the translation from English into another language needs to be compete and accurate.

The Internet, by design, is a decentralized environment. There are very few rules on how it is expected to be used. To a great extent it relies on sets of behavior that are more common practice as opposed to articulated rules. For example, what “rules” exist for tweets on Twitter? What rules exist for Facebook or blog postings. Creating sets of rules will not fly on the Internet because there is no over-arching governing body to enforce any rules. Sure, there are things like Dublin Core with their definitions, but those definitions are left to interpretation, and there are no judges nor courts nor laws determining whether or not any particular application of Dublin Core is “correct”. Only the common use of Dublin Core is correct, and its use is not set in stone.

There are no “should’s” on the Internet. There is only common practice.
With this in mind, it is best for you to work with others both inside and outside your discipline to select one or more ontologies to be used in your linked data. Do not think about this too long nor too hard. It is an never-ending process that is never correct. It is only a process that approximates the best solution.

For simplicity's sake, RDF ontologies are akin to the fields in MARC records or the entities in EAD/XML files. Articulated more accurately, they are the things denoting relationships between subjects and objects in RDF triples. In this light, they are akin to the verbs in all but the most simplistic of sentences. But if they are akin to verbs, then they bring with them all of the nuance and subtlety of human written language. And human written language, in order to be an effective human communications device, comes with two equally important prerequisites: 1) a writer who can speak to an intended audience, and 2) a reader with a certain level of intelligence. A writer who does not use the language of the intended audience speaks to few, and a reader who does not "bring something to the party" goes away with little understanding. Because the effectiveness of every writer is not perfect, and because not every reader comes to the party with a certain level of understanding, written language is imperfect. Similarly, the ontologies of linked data are imperfect. There are no perfect ontologies nor absolutely correct uses of them. There are only best practices and common usages.

This being the case, ontologies still need to be selected in order for linked data to be manifested. What ontologies would you suggest be used when creating linked data for archival descriptions? Here are a few possibilities, listed in no priority order:

* Dublin Core Terms - This ontology is rather bibliographic in nature, and provides a decent framework for describing much of the content of archival descriptions.

* FOAF - Archival collections often originate from individual people. Such is the scope of FOAF, and FOAF is used by a number of other sets of linked data.

* Schema.org - This is an up-and-coming ontology heralded by the 600-pound gorillas in the room -- Google, Microsoft, Yahoo, etc. While the ontology has not been put into practice for very long, it is growing and wide ranging.

* RDF - This ontology is necessary because linked data is manifested as... RDF

* RDFS - This ontology may be necessary because the archival community may be creating some of its own ontologies.

* PROV - for provenance information.

* OWL and SKOS - Both of these ontologies seem to be used to denote relationships between terms in other ontologies. In this way they are used to create classification schemes and thesauri. For example, they allow the implementor to that "creator" in one ontology is the same as "author" in another ontology. Or they allow "country" in one ontology to be denoted as a parent geographic term for "city" in another ontology.

While some or all of these ontologies may be useful for linked data of archival descriptions, what might some other ontologies include? (Remember, it is often "better" to select existing ontologies rather than inventing, unless there is something distinctly unique about a particular domain.) For example, how about an ontology denoting times? Or how about one for places? FOAF is good for people, but what about organizations or institutions?

Another approach is to the data model from other oganizations. Since Europeneana's data is intended to be available as linked data, then it might be a good model to explore — http://pro.europeana.eu/edm-documentation Specifically:

> For the archival community, collection level descriptions such as EAD play a major role.They fit neatly under the EDM,in particular the notion of ore:aggregration allows for describing archival "fonds". The International Council of Archives just started the discussion about a common conceptual model similar to FRBR or the CRM. In the meanwhile, with the CRM historical facts associated with archival contents can be described in more detail than just on the EDM level (Stasinopoulou et. al. 2007). Further, collection-level descriptions in Dublin Core are quite convenient and becoming popular for archival descriptions. — http://ontogenealogy.com/europeana-data-model-edm/

Appendix: Glossary

This is a beginner's glossary to linked data.

- **API** - (see application programmer interface)

- **application programmer interface (API)** - an abstracted set of functions and commands used to get output from remote computer applications. These functions and commands are not necessarily tied to any specific programming language and therefore allow programmers to use a programming language of their choice.

- **content negotiation** - a process whereby a user-agent and HTTP server mutually decide what data format will be exchanged during an HTTP request. In the world of linked data, content negotiation is very important when URIs are requested by user-agents because content negotiation helps determine whether or not HTML or serialized RDF will be returned.

- **extensible markup language (XML)** - a standardized data structure made up of a minimum of rules and can be easily used to represent everything from tiny bits of data to long narrative texts. XML is designed to be read my people as well as computers, but because of this it is often considered verbose, and ironically, difficult to read.

- **HTML** - (see hypertext markup language)

- **HTTP** - (see hypertext transfer protocol)

- **hypertext markup language (HTML)** - an XML-like data structure intended to be rendered by user-agents whose output is for people to read. For the most part, HTML is used to markup text and denote a text's stylistic characteristics such as headers, paragraphs, and list items. It is also used do markup the hypertext links (URLs) between documents.

- **hypertext transfer protocol (HTTP)** - the formal name for the way the World Wide Web operates. It begins with one computer program (a user-agent) requesting content from another computer program (a server) and getting back a response. Once received, the response is formatted for reading by a person or for processing by a computer program. The shape and content of both the request and the response are what make-up the protocol.

- **Javascript object notation (JSON)** - like XML, a data structure enabling allowing arbitrarily large sets of values to associated with an arbitrarily large set of names (variables). JSON was first natively implemented as a part of the Javascript language, but has since become popular in other computer languages.

- **JSON** - (see Javascript object notation)

- **linked data** - the stuff and technical process for making real the ideas behind the Semantic Web. It begins with the creation of serialized RDF and making the serialization available via HTTP. User agents are then expected to harvest the RDF, combine it with other harvested RDF, and ideally use it to bring to light new or existing relationships between real world objects -- people, places, and things -- thus creating and enhancing human knowledge.

- **linked open data** - a qualification of linked data whereby the information being exchanged is expected to be "free" as in gratis.

- **ontology** - a highly structured vocabulary, and in the parlance of linked data, used to denote, describe, and qualify the predicates of RDF triples. Ontologies have been defined for a very wide range of human domains, everything from bibliography (Dublin Core or MODS), to people (FOAF), to sounds (Audio Features).

- **RDF** - (see resource description framework)

- **representational state transfer (REST)** - a process for querying remote HTTP servers and getting back computer-readable results. The process usually employs denoting name-value pairs in a URL and getting back something like XML or JSON.

- **resource description framework** - the conceptual model for describing the knowledge of the Semantic Web. It is rooted in the notion of triples whose subjects and objects are literally linked with other triples through the use of actionable URIs.

- **REST** - (see representational state transfer)

- **Semantic Web** - an idea articulated by Tim Berners Lee whereby human knowledge is expressed in a computer-readable fashion and made available via HTTP so computers can harvest it and bring to light new information or knowledge.

- **serialization** - a manifestation of RDF; one of any number of textual expressions of RDF triples. Examples include but are not limited to RDF/XML, RDFa, N3, and JSON-LD.

- **SPARQL** - (see SPARQL protocol and RDF query language)

- **SPARQL protocol and RDF query language (SPARQL)** - a formal specification for querying and returning results from RDF triple stores. It looks and operates very much like the structured query language (SQL) of relational databases complete with its SELECT, WHERE, and ORDER BY clauses.

- **triple** - the atomistic facts making up RDF. Each fact is akin to a rudimentary sentence with three parts: 1) subject, 2) predicate, and 3) object. Subjects are expected to be URIs. Ideally, objects are URIs as well, but can also be literals (words, phrases, or numbers). Predicates are akin to the verbs in a sentence and they denote a relationship between the subject and object. Predicates are expected to be a member of a formalized ontology.

- **triple store** - a database of RDF triples usually accessible via SPARQL

- **universal resource identifier (URI)** - a unique pointer to a real-world object or a description of an object. In the parlance of linked data, URIs are expected to have the same shape and function as URLs, and if they do, then the URIs are often described as "actionable".

- **universal resource locator (URL)** - an address denoting the location of something on the Internet. These addresses usually specify a protocol (like http), a host (or computer) where the protocol is implemented, and a path (directory and file) specifying where on the computer the item of interest resides.

- **URI** - (see universal resource identifier)

- **URL** - (see universal resource locator)

- **user agent** - this is the formal name for what is commonly called a "Web browser", but Web browsers usually denote applications where people are viewing the results. User agents are usually "Web browsers" whose readers are computer programs.

- **XML** - (see extensible markup language)

For a more complete and exhaustive glossary, see the W3C's Linked Data Glossary. [3]

[3] W3C's Linked Data Glossary can be found at http://www.w3.org/TR/ld-glossary/

Appendix: Use cases

What can you do with linked data once it is created? Here are three use cases:

1. **Do simple publishing** - At its very root, linked data is about making your data available for others to harvest and use. While the “killer linked data application” has seemingly not reared its head, this does not mean you ought not make your data available at linked data. You won’t see the benefits immediately, but sooner or later (less than 5 years from now), you will see your content creeping into the search results of Internet indexes, into the work of both computational humanists and scientists, and into the hands of esoteric hackers creating one-off applications. Internet search engines will create “knowledge graphs”, and they will include links to your content. The humanists and scientists will operate on your data similarly. Both will create visualizations illustrating trends. They will both quantifiably analyze your content looking for patterns and anomalies. Both will probably create network diagrams demonstrating the flow and interconnection of knowledge and ideas through time and space. The humanist might do all this in order to bring history to life or demonstrate how one writer influenced another. The scientist might study ways to efficiently store your data, easily move it around the Internet, or connect it with data set created by their apparatus. The hacker (those are the good guys) will create flashy-looking applications that many will think are weird and useless, but the applications will demonstrate how the technology can be exploited. These applications will inspire others, be here one day and gone the next, and over time, become more useful and sophisticated.

2. **Create a union catalog** - If you make your data available as linked data, and if you find at least one other archive who is making their data available as linked data, then you can find a third somebody who will combine them into a triple store and implement a rudimentary SPARQL interface against the union. Once this is done a researcher could conceivably search the interface for a URI to see what is in both collections. The absolute imperative key to success for this to work is the judicious inclusion of URIs in both data sets. This scenario becomes even more enticing with the inclusion of two additional things. First, the more collections in the triple store the better. You can not have enough collections in the store. Second, the scenario will be even more enticing when each archive publishes their data using similar

ontologies as everybody else. Success does not hinge on similar ontologies, but success is significantly enhanced. Just like the relational databases of today, nobody will be expected to query them using their native query language (SQL or SPARQL). Instead the interfaces will be much more user-friendly. The properties of classes in ontologies will become facets for searching and browsing. Free text as well as fielded searching via drop-down menus will become available. As time goes on and things mature, the output from these interfaces will be increasingly informative, easy-to-read, and computable. This means the output will answer questions, be visually appealing, as well as be available in one or more formats for other computer programs to operate upon.

3. **Tell a story** - You and your hosting institution(s) have something significant to offer. It is not just about you and your archive but also about libraries, museums, the local municipality, etc. As a whole you are a local geographic entity. You represent something significant with a story to tell. Combine your linked data with the linked data of others in your immediate area. The ontologies will be a total hodgepodge, at least at first. Now provide a search engine against the result. Maybe you begin with local libraries or museums. Allow people to search the interface and bring together the content of everybody involved. Do not just provide lists of links in search results, but instead create knowledge graphs. Supplement the output of search results with the linked data from Wikipedia, Flickr, etc. You don't have to be a purist. In a federated search sort of way, supplement the output with content from other data feeds such as (licensed) bibliographic indexes or content harvested from OAI-PMH repositories. Creating these sorts of things on-the-fly will be challenging. On the other hand, you might implement something that is more iterative and less immediate, but more thorough and curated if you were to select a topic or theme of interest, and do your own searching and story telling. The result would be something that is at once a Web page, a document designed for printing, or something importable into another computer program.

Appendix: Scripts

```
#!/usr/bin/perl

# marc2rdf.pl - make MARC records accessible via linked data

# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 5, 2013 - first cut;


# configure
use constant ROOT      => '/disk01/www/html/main/sandbox/liam';
use constant MARC      => ROOT . '/src/marc/';
use constant DATA      => ROOT . '/data/';
use constant PAGES     => ROOT . '/pages/';
use constant MARC2HTML => ROOT . '/etc/MARC21slim2HTML.xsl';
use constant MARC2MODS => ROOT . '/etc/MARC21slim2MODS3.xsl';
use constant MODS2RDF  => ROOT . '/etc/mods2rdf.xsl';
use constant MAXINDEX  => 100;

# require
use IO::File;
use MARC::Batch;
use MARC::File::XML;
use strict;
use XML::LibXML;
use XML::LibXSLT;

# initialize
my $parser = XML::LibXML->new;
my $xslt   = XML::LibXSLT->new;

# process each record in the MARC directory
my @files = glob MARC . "*.marc";
for ( 0 .. $#files ) {

  # re-initialize
  my $marc = $files[ $_ ];
  my $handle = IO::File->new( $marc );
  binmode( STDOUT, ':utf8' );
  binmode( $handle, ':bytes' );
```

```
my $batch  = MARC::Batch->new( 'USMARC', $handle );
$batch->warnings_off;
$batch->strict_off;
my $index = 0;

# process each record in the batch
while ( my $record = $batch->next ) {

  # get marcxml
  my $marcxml =  $record->as_xml_record;
  my $_001    =  $record->field( '001' )->as_string;
  $_001       =~ s/_//;
  $_001       =~ s/ +//;
  $_001       =~ s/-+//;
  print "        marc: $marc\n";
  print "  identifier: $_001\n";
  print "         URI: http://infomotions.com/sandbox/liam/id/$_001\n";

  # re-initialize and sanity check
  my $output = PAGES . "$_001.html";
  if ( ! -e $output or -s $output == 0 ) {

    # transform marcxml into html
    print "        HTML: $output\n";
    my $source     = $parser->parse_string( $marcxml )  or warn $!;
    my $style      = $parser->parse_file( MARC2HTML )   or warn $!;
    my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
    my $results    = $stylesheet->transform( $source )  or warn $!;
    my $html       = $stylesheet->output_string( $results );

    &save( $output, $html );

  }
  else { print "        HTML: skipping\n" }

  # re-initialize and sanity check
  my $output = DATA . "$_001.rdf";
  if ( ! -e $output or -s $output == 0 ) {

    # transform marcxml into mods
    my $source     = $parser->parse_string( $marcxml )  or warn $!;
    my $style      = $parser->parse_file( MARC2MODS )   or warn $!;
```

```
      my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
      my $results    = $stylesheet->transform( $source )  or warn $!;
      my $mods       = $stylesheet->output_string( $results );

      # transform mods into rdf
      print "         RDF: $output\n";
      $source        = $parser->parse_string( $mods )     or warn $!;
      my $style      = $parser->parse_file( MODS2RDF )    or warn $!;
      my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
      my $results    = $stylesheet->transform( $source )  or warn $!;
      my $rdf        = $stylesheet->output_string( $results );

      &save( $output, $rdf );

    }
    else { print "         RDF: skipping\n" }

    # prettify
    print "\n";

    # increment and check
    $index++;
    last if ( $index > MAXINDEX )

  }

}

# done
exit;


sub save {

  open F, ' > ' . shift or die $!;
  binmode( F, ':utf8' );
  print F shift;
  close F;
  return;

}
```

```
#!/usr/bin/perl

# ead2rdf.pl - make EAD files accessible via linked data

# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 6, 2013 - based on marc2linkedata.pl


# configure
use constant ROOT     => '/disk01/www/html/main/sandbox/liam';
use constant EAD      => ROOT . '/src/ead/';
use constant DATA     => ROOT . '/data/';
use constant PAGES    => ROOT . '/pages/';
use constant EAD2HTML => ROOT . '/etc/ead2html.xsl';
use constant EAD2RDF  => ROOT . '/etc/ead2rdf.xsl';
use constant SAXON    => 'java -jar /disk01/www/html/main/sandbox/liam/bin/saxon.jar -
s:##SOURCE## -xsl:##XSL## -o:##OUTPUT##';

# require
use strict;
use XML::XPath;
use XML::LibXML;
use XML::LibXSLT;

# initialize
my $saxon  = '';
my $xsl    = '';
my $parser = XML::LibXML->new;
my $xslt   = XML::LibXSLT->new;

# process each record in the EAD directory
my @files = glob EAD . "*.xml";
for ( 0 .. $#files ) {

  # re-initialize
  my $ead = $files[ $_ ];
  print "         EAD: $ead\n";

  # get the identifier
  my $xpath      = XML::XPath->new( filename => $ead );
  my $identifier = $xpath->findvalue( '/ead/eadheader/eadid' );
  $identifier    =~ s/[^\w ]//g;
```

```
    print "  identifier: $identifier\n";
    print "         URI: http://infomotions.com/sandbox/liam/id/$identifier\n";

    # re-initialize and sanity check
    my $output = PAGES . "$identifier.html";
    if ( ! -e $output or -s $output == 0 ) {

      # transform marcxml into html
      print "        HTML: $output\n";
      my $source     = $parser->parse_file( $ead )  or warn $!;
      my $style      = $parser->parse_file( EAD2HTML )   or warn $!;
      my $stylesheet = $xslt->parse_stylesheet( $style )  or warn $!;
      my $results    = $stylesheet->transform( $source )  or warn $!;
      my $html       = $stylesheet->output_string( $results );

      &save( $output, $html );

    }
    else { print "        HTML: skipping\n" }

    # re-initialize and sanity check
    my $output = DATA . "$identifier.rdf";
    if ( ! -e $output or -s $output == 0 ) {

      # create saxon command, and save rdf
      print "         RDF: $output\n";
      $saxon  =  SAXON;
      $xsl    =  EAD2RDF;
      $saxon  =~ s/##SOURCE##/$ead/e;
      $saxon  =~ s/##XSL##/$xsl/e;
      $saxon  =~ s/##OUTPUT##/$output/e;
      system $saxon;

    }
    else { print "         RDF: skipping\n" }

    # prettify
    print "\n";

  }

  # done
```

```
exit;


sub save {

  open F, ' > ' . shift or die $!;
  binmode( F, ':utf8' );
  print F shift;
  close F;
  return;

}
```

```
#!/usr/bin/perl

# store-make.pl - simply initialize an RDF triple store
# Eric Lease Morgan <eric_morgan@infomotions.com>
#
# December 14, 2013 - after wrestling with wilson for most of the day


# configure
use constant ETC => '/disk01/www/html/main/sandbox/liam/etc/';

# require
use strict;
use RDF::Redland;

# sanity check
my $db = $ARGV[ 0 ];
if ( ! $db ) {

  print "Usage: $0 <db>\n";
  exit;

}

# do the work; brain-dead
my $etc = ETC;
my $store = RDF::Redland::Storage->new( 'hashes', $db, "new='yes', hash-type='bdb', 
dir='$etc'" );
die "Unable to create store ($!)" unless $store;
my $model = RDF::Redland::Model->new( $store, '' );
die "Unable to create model ($!)" unless $model;

# "save"
$store = undef;
$model = undef;

# done
exit;
```

```
#!/usr/bin/perl

# store-add.pl - add items to an RDF triple store
# Eric Lease Morgan <eric_morgan@infomotions.com>
#
# December 14, 2013 - after wrestling with wilson for most of the day


# configure
use constant ETC => '/disk01/www/html/main/sandbox/liam/etc/';

# require
use strict;
use RDF::Redland;

# sanity check #1 - command line arguments
my $db   = $ARGV[ 0 ];
my $file = $ARGV[ 1 ];
if ( ! $db or ! $file ) {

  print "Usage: $0 <db> <file>\n";
  exit;

}

# sanity check #2 - store exists
die "Error: po2s file not found. Make a store?\n" if ( ! -e ETC . $db . '-po2s.db' );
die "Error: so2p file not found. Make a store?\n" if ( ! -e ETC . $db . '-so2p.db' );
die "Error: sp2o file not found. Make a store?\n" if ( ! -e ETC . $db . '-sp2o.db' );

# open the store
my $etc = ETC;
my $store = RDF::Redland::Storage->new( 'hashes', $db, "new='no', hash-type='bdb',
dir='$etc'" );
die "Error: Unable to open store ($!)" unless $store;
my $model = RDF::Redland::Model->new( $store, '' );
die "Error: Unable to create model ($!)" unless $model;

# sanity check #3 - file exists
die "Error: $file not found.\n" if ( ! -e $file );

# parse a file and add it to the store
```

```
my $uri    = RDF::Redland::URI->new( "file:$file" );
my $parser = RDF::Redland::Parser->new( 'rdfxml', 'application/rdf+xml' );
die "Error: Failed to find parser ($!)\n" if ( ! $parser );
my $stream = $parser->parse_as_stream( $uri, $uri );
my $count  = 0;
while ( ! $stream->end ) {

  $model->add_statement( $stream->current );
  $count++;
  $stream->next;

}

# echo the result
warn "Namespaces:\n";
my %namespaces = $parser->namespaces_seen;
while ( my ( $prefix, $uri ) = each %namespaces ) {

  warn " prefix: $prefix\n";
  warn '    uri: ' . $uri->as_string . "\n";
  warn "\n";

}
warn "Added $count statements\n";

# "save"
$store = undef;
$model = undef;

# done
exit;
```

```
#!/usr/bin/perl

# store-search.pl - query a triple store
# Eric Lease Morgan <eric_morgan@infomotions.com>

# December 14, 2013 - after wrestling with wilson for most of the day


# configure
use constant ETC => '/disk01/www/html/main/sandbox/liam/etc/';
my %namespaces = (

  "crm"       => "http://erlangen-crm.org/current/",
  "dc"        => "http://purl.org/dc/elements/1.1/",
  "dcterms"   => "http://purl.org/dc/terms/",
  "event"     => "http://purl.org/NET/c4dm/event.owl#",
  "foaf"      => "http://xmlns.com/foaf/0.1/",
  "lode"      => "http://linkedevents.org/ontology/",
  "lvont"     => "http://lexvo.org/ontology#",
  "modsrdf"   => "http://simile.mit.edu/2006/01/ontologies/mods3#",
  "ore"       => "http://www.openarchives.org/ore/terms/",
  "owl"       => "http://www.w3.org/2002/07/owl#",
  "rdf"       => "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
  "rdfs"      => "http://www.w3.org/2000/01/rdf-schema#",
  "role"      => "http://simile.mit.edu/2006/01/roles#",
  "skos"      => "http://www.w3.org/2004/02/skos/core#",
  "time"      => "http://www.w3.org/2006/time#",
  "timeline"  => "http://purl.org/NET/c4dm/timeline.owl#",
  "wgs84_pos" => "http://www.w3.org/2003/01/geo/wgs84_pos#"

);

# require
use strict;
use RDF::Redland;

# sanity check #1 - command line arguments
my $db    = $ARGV[ 0 ];
my $query = $ARGV[ 1 ];
if ( ! $db or ! $query ) {

  print "Usage: $0 <db> <query>\n";
  exit;

}

# sanity check #2 - store exists
die "Error: po2s file not found. Make a store?\n" if ( ! -e ETC . $db . '-po2s.db' );
die "Error: so2p file not found. Make a store?\n" if ( ! -e ETC . $db . '-so2p.db' );
die "Error: sp2o file not found. Make a store?\n" if ( ! -e ETC . $db . '-sp2o.db' );

# open the store
my $etc = ETC;
my $store = RDF::Redland::Storage->new( 'hashes', $db, "new='no', hash-type='bdb',
dir='$etc'" );
die "Error: Unable to open store ($!)" unless $store;
my $model = RDF::Redland::Model->new( $store, '' );
die "Error: Unable to create model ($!)" unless $model;

# search
#my $sparql  = RDF::Redland::Query->new( "CONSTRUCT { ?a ?b ?c } WHERE { ?a ?b ?c }",
undef, undef, "sparql" );
my $sparql = RDF::Redland::Query->new( "PREFIX modsrdf: <http://simile.mit.edu/
2006/01/ontologies/mods3#>\nSELECT ?a ?b ?c WHERE {  ?a  modsrdf:$query ?c }", undef,
undef, 'sparql' );
my $results = $model->query_execute( $sparql );
print $results->to_string;

# done
exit;
```

```perl
#!/usr/bin/perl

# store-dump.pl - output the content of store as RDF/XML
# Eric Lease Morgan <eric_morgan@infomotions.com>
#
# December 14, 2013 - after wrestling with wilson for most of the day


# configure
use constant ETC => '/disk01/www/html/main/sandbox/liam/etc/';

# require
use strict;
use RDF::Redland;

# sanity check #1 - command line arguments
my $db  = $ARGV[ 0 ];
my $uri = $ARGV[ 1 ];
if ( ! $db ) {

  print "Usage: $0 <db> <uri>\n";
  exit;

}

# sanity check #2 - store exists
die "Error: po2s file not found. Make a store?\n" if ( ! -e ETC . $db . '-po2s.db' );
die "Error: so2p file not found. Make a store?\n" if ( ! -e ETC . $db . '-so2p.db' );
die "Error: sp2o file not found. Make a store?\n" if ( ! -e ETC . $db . '-sp2o.db' );

# open the store
my $etc = ETC;
my $store = RDF::Redland::Storage->new( 'hashes', $db, "new='no', hash-type='bdb',
dir='$etc'" );
die "Error: Unable to open store ($!)" unless $store;
my $model = RDF::Redland::Model->new( $store, '' );
die "Error: Unable to create model ($!)" unless $model;

# do the work
my $serializer = RDF::Redland::Serializer->new;
print $serializer->serialize_model_to_string( RDF::Redland::URI->new, $model );
```

```
# done
exit;
```

```
#!/usr/bin/perl

# sparql.pl - a brain-dead, half-baked SPARQL endpoint

# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 15, 2013 - first investigations


# require
use CGI;
use CGI::Carp qw( fatalsToBrowser );
use RDF::Redland;
use strict;

# initialize
my $cgi   = CGI->new;
my $query = $cgi->param( 'query' );

if ( ! $query ) {

  print $cgi->header;
  print &home;

}

else {

  # open the store for business
  my $store = RDF::Redland::Storage->new( 'hashes', 'store', "new='no', hash-
type='bdb', dir='/disk01/www/html/main/sandbox/liam/etc'" );
  my $model = RDF::Redland::Model->new( $store, '' );

  # search
  my $results = $model->query_execute( RDF::Redland::Query->new( $query, undef, undef,
'sparql' ) );

  # return the results
  print $cgi->header( -type => 'application/xml' );
  print $results->to_string;

}
```

```
# done
exit;


sub home {

  # create a list namespaces
  my $namespaces = &namespaces;
  my $list       = '';
  foreach my $prefix ( sort keys $namespaces ) {

    my $uri = $$namespaces{ $prefix };
    $list .= $cgi->li( "$prefix - " . $cgi->a( { href=> $uri, target => '_blank' },
$uri ) );

  }
  $list = $cgi->ol( $list );

  # return a home page
  return <<EOF
<html>
<head>
<title>LiAM SPARQL Endpoint</title>
</head>
<body style='margin: 7%'>
<h1>LiAM SPARQL Endpoint</h1>
<p>This is a brain-dead and half-baked SPARQL endpoint to a subset of LiAM linked
data. Enter a query, but there is the disclaimer. Errors will probably happen because
of SPARQL syntax errors. Remember, the interface is brain-dead. Your milage <em>will</
em> vary.</p>
<form method='GET' action='./'>
<textarea style='font-size: large' rows='5' cols='65' name='query' />
PREFIX hub:<http://data.archiveshub.ac.uk/def/>
SELECT ?uri
WHERE { ?uri ?o hub:FindingAid }
</textarea><br />
<input type='submit' value='Search' />
</form>
<p>Here are a few sample queries:</p>
<ul>
  <li>Find all triples with RDF Schema labels - <code><a href="http://infomotions.com/
sandbox/liam/sparql/?query=PREFIX+rdf%3A%3Chttp%3A%2F%2Fwww.w3.org%2F2000%2F01%2Frdf-
```

```
schema%23%3E%0D%0ASELECT+*+WHERE+%7B+%3Fs+rdf%3Alabel+%3Fo+%7D%0D%0A">PREFIX
rdf:&lt;http://www.w3.org/2000/01/rdf-schema#&gt; SELECT * WHERE { ?s rdf:label ?o }</
a></code></li>
  <li>Find all items with MODS subjects - <code><a href='http://infomotions.com/
sandbox/liam/sparql/?query=PREFIX+mods%3A%3Chttp%3A%2F%2Fsimile.mit.edu
%2F2006%2F01%2Fontologies%2Fmods3%23%3E%0D%0ASELECT+*+WHERE+%7B+%3Fs+mods%3Asubject+
%3Fo+%7D'>PREFIX mods:&lt;http://simile.mit.edu/2006/01/ontologies/mods3#&gt; SELECT *
WHERE { ?s mods:subject ?o }</a></code></li>
  <li>Find every unique predicate - <code><a href="http://infomotions.com/sandbox/
liam/sparql/?query=SELECT+DISTINCT+%3Fp+WHERE+%7B+%3Fs+%3Fp+%3Fo+%7D">SELECT
DISTINCT ?p WHERE { ?s ?p ?o }</a></code></li>
  <li>Find everything - <code><a href="http://infomotions.com/sandbox/liam/sparql/?
query=SELECT+*+WHERE+%7B+%3Fs+%3Fp+%3Fo+%7D">SELECT * WHERE { ?s ?p ?o }</a></code></
li>
  <li>Find all classes - <code><a href="http://infomotions.com/sandbox/liam/sparql/?
query=SELECT+DISTINCT+%3Fclass+WHERE+%7B+%5B%5D+a+%3Fclass+%7D+ORDER+BY+
%3Fclass">SELECT DISTINCT ?class WHERE { [] a ?class } ORDER BY ?class</a></code></li>
  <li>Find all properties - <code><a href="http://infomotions.com/sandbox/liam/
sparql/?query=SELECT+DISTINCT+%3Fproperty%0D%0AWHERE+%7B+%5B%5D+%3Fproperty+%5B%5D+%7D
%0D%0AORDER+BY+%3Fproperty">SELECT DISTINCT ?property WHERE { [] ?property [] } ORDER
BY ?property</a></code></li>
  <li>Find URIs of all finding aids - <code><a href="http://infomotions.com/sandbox/
liam/sparql/?query=PREFIX+hub%3A%3Chttp%3A%2F%2Fdata.archiveshub.ac.uk%2Fdef%2F%3E
+SELECT+%3Furi+WHERE+%7B+%3Furi+%3Fo+hub%3AFindingAid+%7D">PREFIX hub:&lt;http://
data.archiveshub.ac.uk/def/&gt; SELECT ?uri WHERE { ?uri ?o hub:FindingAid }</a></
code></li>
  <li>Find URIs of all MARC records - <code><a href="http://infomotions.com/sandbox/
liam/sparql/?query=PREFIX+mods%3A%3Chttp%3A%2F%2Fsimile.mit.edu
%2F2006%2F01%2Fontologies%2Fmods3%23%3E+SELECT+%3Furi+WHERE+%7B+%3Furi+%3Fo+mods
%3ARecord+%7D%0D%0A%0D%0A%0D%0A">PREFIX mods:&lt;http://simile.mit.edu/2006/01/
ontologies/mods3#&gt; SELECT ?uri WHERE { ?uri ?o mods:Record }</a></code></li>
  <li>Find all URIs of all collections - <code><a href="http://infomotions.com/
sandbox/liam/sparql/?query=PREFIX+mods%3A%3Chttp%3A%2F%2Fsimile.mit.edu
%2F2006%2F01%2Fontologies%2Fmods3%23%3E%0D%0APREFIX+hub%3A%3Chttp%3A%2F
%2Fdata.archiveshub.ac.uk%2Fdef%2F%3E%0D%0ASELECT+%3Furi+WHERE+%7B+%7B+%3Furi+%3Fo+hub
%3AFindingAid+%7D+UNION+%7B+%3Furi+%3Fo+mods%3ARecord+%7D+%7D%0D%0AORDER+BY+%3Furi%0D
%0A">PREFIX mods:&lt;http://simile.mit.edu/2006/01/ontologies/mods3#&gt; PREFIX
hub:&lt;http://data.archiveshub.ac.uk/def/&gt; SELECT ?uri WHERE { { ?uri ?o
hub:FindingAid } UNION { ?uri ?o mods:Record } } ORDER BY ?uri</a></code></li>
</ul>
<p>This is a list of ontologies (namespaces) used in the triple store as predicates:</
p>
```

```
$list
<p>For more information about SPARQL, see:</p>
<ol>
  <li><a href="http://www.w3.org/TR/rdf-sparql-query/" target="_blank">SPARQL Query
Language for RDF</a> from the W3C</li>
  <li><a href="http://en.wikipedia.org/wiki/SPARQL" target="_blank">SPARQL</a> from
Wikipedia</li>
</ol>
<p>Source code -- <a href="http://infomotions.com/sandbox/liam/bin/
sparql.pl">sparql.pl</a> -- is available online.</p>
<hr />
<p>
<a href="mailto:eric_morgan\@infomotions.com">Eric Lease Morgan &lt;eric_morgan
\@infomotions.com&gt;</a><br />
January 6, 2014
</p>
</body>
</html>
EOF

}


sub namespaces {

  my %namespaces = (

    "crm"       => "http://erlangen-crm.org/current/",
    "dc"        => "http://purl.org/dc/elements/1.1/",
    "dcterms"   => "http://purl.org/dc/terms/",
    "event"     => "http://purl.org/NET/c4dm/event.owl#",
    "foaf"      => "http://xmlns.com/foaf/0.1/",
    "lode"      => "http://linkedevents.org/ontology/",
    "lvont"     => "http://lexvo.org/ontology#",
    "modsrdf"   => "http://simile.mit.edu/2006/01/ontologies/mods3#",
    "ore"       => "http://www.openarchives.org/ore/terms/",
    "owl"       => "http://www.w3.org/2002/07/owl#",
    "rdf"       => "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
    "rdfs"      => "http://www.w3.org/2000/01/rdf-schema#",
    "role"      => "http://simile.mit.edu/2006/01/roles#",
    "skos"      => "http://www.w3.org/2004/02/skos/core#",
    "time"      => "http://www.w3.org/2006/time#",
    "timeline"  => "http://purl.org/NET/c4dm/timeline.owl#",
    "wgs84_pos" => "http://www.w3.org/2003/01/geo/wgs84_pos#"

  );

  return \%namespaces;

}
```

```
package Apache2::LiAM::Dereference;

# Dereference.pm - Redirect user-agents based on value of URI.

# Eric Lease Morgan <eric_morgan@infomotions.com>
# December 7, 2013 - first investigations; based on Apache2::Alex::Dereference


# configure
use constant PAGES => 'http://infomotions.com/sandbox/liam/pages/';
use constant DATA  => 'http://infomotions.com/sandbox/liam/data/';

# require
use Apache2::Const -compile => qw( OK );
use CGI;
use strict;

# main
sub handler {

  # initialize
  my $r   = shift;
  my $cgi = CGI->new;
  my $id  = substr( $r->uri, length $r->location );

  # wants RDF
  if ( $cgi->Accept( 'text/html' )) {

    print $cgi->header( -status => '303 See Other',
    -Location => PAGES . $id . '.html',
    -Vary     => 'Accept' )

  }

  # give them RDF
  else {

    print $cgi->header( -status => '303 See Other',
    -Location      => DATA . $id . '.rdf',
    -Vary          => 'Accept',
    "Content-Type" => 'application/rdf+xml' )
  }

  # done
  return Apache2::Const::OK;

}

1; # return true or die
```