BUILDING A LINKED DATA APP

Libraries, Tools, and Tips for Making a Working App

THINGS TO KNOW ABOUT MAKING A LINKED DATA APP

- Code libraries you can use
- Utilities that can help think through what code you need
- Linked data sources that might prove valuable
- Finally, putting it all together with a small demo app

FIRST STEPS: CHOOSING YOUR LIBRARY

- Almost any language has an RDF library...
- Redland, Raptor, and Rasqal (C, with bindings for many others)
- Jena (Java)
- RDFLib (Python)
- ARC (PHP)

REDLAND, RAPTOR, RASQAL

- Probably the most fully featured library set, and one of the longest in development.
  - Redland handles storage, retrieval.
  - Raptor parses RDF.
  - Rasqal handles SPARQL queries (but not SPARQL/Update!).
- Straightforward C API.
- Bindings for (among others) Perl, Python, PHP, Ruby, C#, Objective-C

JENA

- Jena is the standard Java RDF library.
- Supports SPARQL and SPARUL with ARQ library.
- More oriented towards reasoning.
- Pellet, an OWL reasoner, depends on Jena and can be used from other Java code.
- D2RQ, a relational database to semantic web mapping engine also uses Jena.

RDFLIB

- Python library with native-components
- Somewhat easier to use than Redland bindings
- Doesn't natively support SPARQL, but can query SPARQL Endpoints using it and another library.
- Supports several backends, including Redland.

ARC2

- A PHP-native library that supports MySQL backend.
- Can just drop in and import into existing PHP code without compilation (like Redland).
- Supports SPARQL, SPARQL+, and use of endpoints as alternate storage.
- Can be slightly constricting with advanced SPARQL queries.
- My example app will use this.

HELPFUL UTILITIES AND TIPS

- The W3C RDF Validation service is useful to check RDF/XML.
- Use cwm or rapper (part of Raptor) to validate your N3 syntax.
- Use a Linked Data browser, like Tabulator, to check your data.
- Run your SPARQL queries against the endpoint to make sure they work.
- Make sure you clean your data! (Especially from DBPedia!)

HELPFUL LINKED DATA SOURCES

- sameas.org: Provides comprehensive owl:sameAs links.
- DBPedia: Pretty much any Wikipedia topic has a DBPedia resource, with most of the data from templates extracted.
- Geonames: Invaluable pointers and descriptors for geographic locations.
- prefix.cc: Common prefixes for vocabularies.
- There are a number of sources that tap into other domains. Feel free to ask.

A SAMPLE APP

- MusicBrainz is a useful music database with linked data endpoints.

- While it now has tags for artists, albums, and labels, it doesn't have explicit genres, and is less detailed.

- DBPedia has useful groupings of genres, and links to artists, but lacks MusicBrainz's breadth of albums.

- Why not link the two?

THE PROBLEM: PART A

- Use DBTune, DBPedia, and any other sources you can think of to get a list of artists on DBTune (and their MusicBrainz ID, see **`mo:musicbrainz`**) when provided a DBPedia genre.

- List genre URIs for an artist when provided a MusicBrainz ID.

THE PROBLEM: PART B

- Extend the program to flesh out the "artist" and "genre" descriptions.
- Artists should have a short description, the instruments they play, the actual NAMES of the genres, birth and/or death dates, and the years they were active.
- Artists should also list their albums, with release dates, number of tracks, and Amazon links.
- Genres should have the years they were popular, a short description, and link to any related genres (e.g. subgenres, derivatives, stylistic origins, and those genres for which it is a subgenre/derivative/stylistic origin)

THE PROBLEM: PART C

- Surprise me!

- Do something else exciting with your program when I provide an artist or genre.

STEP 1: ARTIST PAGES

- Started by pulling information from DBTune (which hosts a MusicBrainz endpoint)

- Also queried for albums.

- http://www.telegraphis.net/demoapps/music/artist1.php

- Source: http://www.telegraphis.net/demoapps/music/artist1.phps

STEP 2: LINK THAT DATA!

- DBTune's MusicBrainz resource has owl:sameAs links to DBPedia.
- Use those to query DBPedia for more information, like genres.
- http://www.telegraphis.net/demoapps/music/artist2.php
- Source: http://www.telegraphis.net/demoapps/music/artist2.phps

STEP 3: BRING ON THE GENRES

- Now that we have genres, we can build genre pages.
- Modified artist pages to link to genre pages.
- Constructed genre pages by querying DBPedia for data and artists.
- Then used sameas.org to get DBTune URIs (could have queried DBTune to do the same)
- Queried DBTune for the number of albums per artist.

STEP 3: BRING ON THE GENRES

- http://www.telegraphis.net/demoapps/music/artist3.php
  http://www.telegraphis.net/demoapps/music/genre1.php

- Source:
  http://www.telegraphis.net/demoapps/music/artist3.phps
  http://www.telegraphis.net/demoapps/music/genre1.phps

AND ONWARD!

- Obvious next steps include styling and some other data fields.
- Cleaning up bad data...
- Final result at:
  http://www.telegraphis.net/demoapps/music/artist.php
  http://www.telegraphis.net/demoapps/music/genre.php
- Source:
  http://www.telegraphis.net/demoapps/music/artist.phps
  http://www.telegraphis.net/demoapps/music/genre.phps

SO HOW LONG DID IT TAKE?

- Relatively straightforward to improve from here...

- This only took me 3 hours to put together through the third revision, including time looking up documentation.

- Probably faster (hour and a half?) for PHP programmers (I haven't done so for some time!)

LIBRARY LINKS

- Redland: http://www.librdf.org/
- Jena: http://jena.sourceforge.net/
- RDFLib: http://www.rdflib.net/
  (http://sparql-wrapper.sourceforge.net/ for a Python-based SPARQL endpoint wrapper)
- ARC2: http://arc2.semsol.net/
- ActiveRDF (for Ruby): http://www.activerdf.org/

MORE HELPFUL LINKS

- cwm: http://www.w3.org/2000/10/swap/doc/cwm.html
- W3C RDF/XML Validator: http://www.w3.org/RDF/Validator/
- Tabulator is available on the course resource page: http://dig.csail.mit.edu/2010/LinkedData/res.html
- IRC channel #swig on irc.freenode.net has plenty of semantic web gurus from around the world to help out with questions
- ESW Wiki: http://esw.w3.org/topic/FrontPage

COMMON ONTOLOGIES/VOCABULARIES

- ESW Wiki has some: http://esw.w3.org/topic/TaskForces/CommunityProjects/LinkingOpenData/CommonVocabularies

- I've collected a few useful ones: http://delicious.com/pipian/ontology

- A little outdated, but perhaps still useful is SchemaWeb: http://www.schemaweb.info/

- And don't be afraid to look at the data sources themselves and visit the namespace URIs of unfamiliar vocabularies...

DATASOURCES THAT MIGHT PROVE USEFUL:

- DBTune MusicBrainz Data: http://dbtune.org/musicbrainz/sparql
- DBPedia: http://dbpedia.org/sparql
- Other DBTune data (Jamendo, Magnatune, last.fm scrobbles, etc.): http://dbtune.org/
- Geonames: http://www.geonames.org/export/ws-overview.html
- Links to more sources on the Linked Data "map": http://www4.wiwiss.fu-berlin.de/bizer/pub/lod-datasets_2009-03-05.html

QUESTIONS?

GOOD LUCK!