Pode - LinkedNonFiction - Documentation

Website: http://www.bibpode.no
Blog: http://www.bibpode.no/blogg
Twitter: @podeprosjekt
Facebook: http://www.facebook.com/pages/Pode/133789476637255
GitHub: https://github.com/pode
E-mail: pode@deichman.no

1. LinkedNonFiction

Dataset

The original dataset is an export from the Oslo Public Library catalogue, produced by a CCL-query done by PHP/YAZ. The query asks for all documents owned by the Multilingual library, that have literary format code 0 (non-fiction), and that does not contain music. The returned set consists of MARC records for ca. 16 000 documents in 94 different languages, expressed as XML.

The MARC-records were converted to RDF turtle format by XSLT, according to a vocabulary, mainly based on the Bibo bibliographic ontology.

- Vocabulary: https://github.com/pode/LinkedNonFiction/blob/master/marc2rdf.rdf
- XSLT: https://github.com/pode/LinkedNonFiction/blob/master/marcslim2n3.xsl

Expressing classification codes

The idea is to use multilingual Dewey class labels from http://dewey.info/ to browse the collection. This dataset represents the top three levels of the Dewey decimal classification. Most books will be classified at a more specialiced level than this, so in addition to the fulll DDC code found in MARC field 082, we also generated properties to connect documents to the correct top levels. A document about bats, classified with Dewey code 599.4, will have the following statements for Dewey class connection:

```
<document> pode:ddkFirst instance:ddk_5 ; (Science)
pode:ddkSecond instance:ddk_59 ; (Zoology)
pode:ddkFirst instance:ddk_599 ; (Mammals)
pode:ddkFirst instance:ddk_5994 . (Bats)
```

Expressing subjects

Catalogue records also contain Dewey codes that describe subject headings. Since subject headings are expressed as instances of skos:Concept, we use the property sub:classification from the Sublima vocabulary. Subject heading strings are expressed as concepts on their own, with broader/narrower relation to the subjects that make up the string:

```
topic:poland_history a skos:Concept ;
skos:prefLabel "Poland - History" ;
skos:broader topic:poland ;
skos:broader topic:history ;
sub:classification instance:ddk_9438 .
```

When a catalogue record uses MARC field 600 to state that a person is the topic of the document, we will create both a skos:Concept and a foaf:Person. They will be connected like this:

```
<document> dct:subject topic:leonardo_da_vinci .

topic:leonardo_da_vinci a skos:Concept;
skos:prefLabel "Leonardo da Vinci" ;
```

```
foaf:focus person:Leonardo_da_Vinci .

person:Leonardo_da_Vinci a foaf:Person ;
foaf:name "Leonardo da Vinci" .
```

Nudging language data

Using language instances from the Lexvo vocabulary turned out to be a little less straightforward than expected. There is, as far as we could see, no such thing as a SPARQL endpoint for Lexvo data. I order to query the data in our web application, we therefore had to import these data into our own triplestore. Furthermore, it turned out that the data weren't as complete as we had hoped. Several languges lacked a label in Norwegian. This turned out to cause a problem in our web application, since some SPARQL queries failed to provide an answer. We solved this by adding the missing labels to our local version of the Lexvo dataset, using the SPARQL Insert commando.

2. LinkedNonFictionWeb

LinkedNonFictionWeb is a web-based display of the data created by LinkedNonFiction, mashed up with data from a couple of other sources.

- Demo: http://bibpode.no/linkednonfiction/
- Source code: http://github.com/pode/LinkedNonFictionWeb

Data sources

- LinkedNonFiction - data stored in an ARC triplestore
  - SPARQL endpoint: http://bibpode.no/rdfstore/endpoint.php
- dewey.info
  - SPARQL endpoint: http://dewey.info/sparql.php
- Lexvo.org
  - Individual RDF files are available at e.g. http://www.lexvo.org/data/iso639-3/dan
- Sources for cover images
  - Bokkilden - http://www.bokkilden.no/SamboWeb/partneradmin.do?rom=MP
  - Open Library - http://openlibrary.org/dev/docs/api/covers

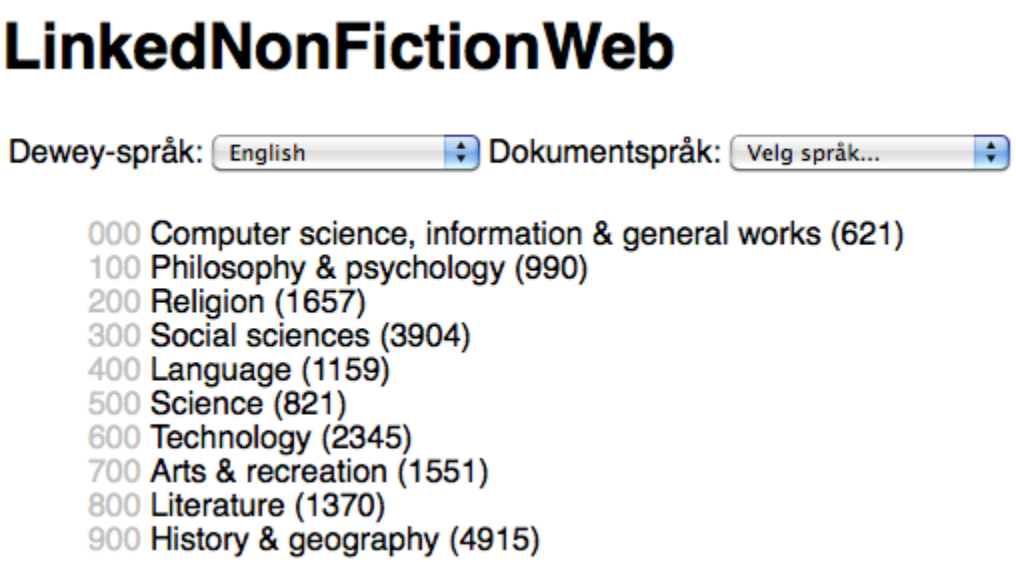
Architectural outline

LinkedAuthorsWeb is based around an HTML page, enhanced with CSS. Data is fetched from different sources and presented to the user with AJAX-techniques, utilising the jQuery JavaScript library.

- Sources that can return data in JSONP format are queried directly from JavaScript.
- Sources that can only return JSON need to have their data passed through a local proxy (written in PHP) to avoid cross-site scripting limitations in browsers.

Logic and queries

The initial view of LinkedNonFictionWeb consists of two drop-down menus and the top level Dewey categories in English:



Dewey languages

The first drop-down contains the languages that are available from dewey.info. These are requested by the main JavaScript file, LinkedNonFictionWeb.js, and retrieved with the following query, in proxy.php:

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT DISTINCT ?lang WHERE {
?toplevel skos:hasTopConcept ?concept .
?concept dct:language ?lang .
}
```

This returns a list of the available languages in the form of two letter codes. These codes are “translated” into labels by fetching RDF files from Lexvo and parsing out the name of the language in the language itself:

```
function lexvo2name($code, $lexvouri) {

    // We need to massage the URL to get it in a format Lexvo likes
    $url = preg_replace('/\/id\//', '/data/', $lexvouri);
```

```
    $url = preg_replace('/lexvo/', 'www.lexvo', $url);

    // create curl resource
    $ch = curl_init($url);
    curl_setopt($ch, CURLOPT_HEADER, true);
    //return the transfer as a string
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
    curl_setopt($ch, CURLOPT_HTTPHEADER, array (
                "Accept: application/rdf+xml"
      ));
    curl_setopt($ch, CURLOPT_FAILONERROR, true);
    // Get the contents of the page
    $rdf = curl_exec($ch);
    // close curl resource to free up system resources
    curl_close($ch);

    // dewey.info uses "no" for Norwegian, the two letter code used in
    // Lexvo is "nb" or "nn", so we translate to "nb"
    if ($code == 'no') {
       $code = 'nb';
    }

    preg_match('/<rdfs:label rdf:datatype="xsd:string" xml:lang="' .
$code . '">(.*)</', $rdf, $match);
    return $match[1];
}
```

(It would of course have been better to get these labels from the Lexvo data we store in our own ARC triplestore, but this function was written before the Lexvo data were imported, and has not been updated to use them.)

The codes and language labels are returned to LinkedNonFictionWeb.js as JSON and the drop-down menu is assembled.

Top Dewey concepts

When the Dewey language drop-down is displayed, the next stage of the rendering is to display the top Dewey concepts in English. These are retrieved from dewey.info with this SPARQL query:

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT DISTINCT ?concept ?notation ?label WHERE {
?toplevel skos:hasTopConcept ?concept .
?concept skos:notation ?notation .
?concept skos:prefLabel ?label .
?concept dct:language "en" .
```

```
}
```
If at any time another language is chosen from the Dewey language drop-down, a similar query is executed, with the chosen language code substituted for “en”, and the top concepts for that language is displayed.

Hit-counts

For every Dewey concept listed, the number of hits related to the Dewey number are shown in parenthesis. The counts are retrieved with queries similar to this:

```
PREFIX pode: <http://www.bibpode.no/vocabulary#>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT COUNT(?record) AS ?count WHERE {
?record pode:ddkThird <http://www.bibpode.no/instance/DDK_001> .
}
```

If a language is chosen from the document languages drop-down, the hits are re-counted with queries that are modified to limit the count to hits in that language:

```
PREFIX pode: <http://www.bibpode.no/vocabulary#>
PREFIX dct: <http://purl.org/dc/terms/>
SELECT COUNT(?record) AS ?count WHERE {
?record pode:ddkThird <http://www.bibpode.no/instance/DDK_001> .
?record dct:language <http://lexvo.org/id/iso639-3/amh> .
}
```

Document languages

The second drop-down menu contains a list of all the languages that are used in the documents that we have metadata about in the triplestore. The list is retrieved with this query:

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX pode: <http://www.bibpode.no/vocabulary#>
SELECT DISTINCT ?language ?langLabel WHERE {
?record dct:source pode:dfb_fagposter .
?record dct:language ?language .
?language rdfs:label ?langLabel .
FILTER langMatches( lang(?langLabel), "nb" )
} ORDER BY ?langLabel
```

If a language is chosen from this list, the hit counts displayed along with the Dewey concepts are re-calculated, as described above.

Narrower concepts

Clicking on one of the top Dewey concepts displays the narrower concepts on the second level. Clicking on a concept on the second level displays the narrower concepts on the third level. This is done with queries like this:

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX dct: <http://purl.org/dc/terms/>
```

```
SELECT DISTINCT ?narrower ?notation ?label WHERE {
<http://dewey.info/class/7/2009/08/about.pt> skos:narrower ?narrower .
?narrower skos:notation ?notation .
?narrower skos:prefLabel ?label .
?narrower dct:language "pt" .
}
```

When a new level of Dewey concepts is displayed, the hit counts are also fetched as described above.

Hit list

When one of the Dewey concepts on the third and last level is clicked, a search is carried out for that concept and a list of results displayed. The search looks like this:

```
PREFIX pode: <http://www.bibpode.no/vocabulary#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?record ?responsibility ?title ?formatlabel ?issued ?
langlabel WHERE {
?record dct:source pode:dfb_fagposter ;
pode:ddkThird <http://www.bibpode.no/instance/DDK_001> .
OPTIONAL { ?record dct:title ?title . }
OPTIONAL { ?record pode:responsibility ?responsibility . }
OPTIONAL { ?record dct:issued ?issued . }
OPTIONAL {
  ?record dct:format ?format .
  ?format rdfs:label ?formatlabel .
}
OPTIONAL {
  ?record dct:language ?language .
  ?language rdfs:label ?langlabel .
}
FILTER langMatches( lang(?langlabel), "nb" )
} GROUP BY ?record ORDER BY DESC(?issued) ?record
```

If a language has been chosen from the document language drop-down, the search is limited to the chosen language by adding one more constrint to the query:

```
PREFIX pode: <http://www.bibpode.no/vocabulary#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT ?record ?responsibility ?title ?formatlabel ?issued ?
langlabel WHERE {
?record dct:source pode:dfb_fagposter ;
pode:ddkThird <http://www.bibpode.no/instance/DDK_001> .
?record dct:language <http://lexvo.org/id/iso639-3/eng> .
OPTIONAL { ?record dct:title ?title . }
OPTIONAL { ?record pode:responsibility ?responsibility . }
```

```
OPTIONAL { ?record dct:issued ?issued . }
OPTIONAL {
  ?record dct:format ?format .
  ?format rdfs:label ?formatlabel .
}
OPTIONAL {
  ?record dct:language ?language .
  ?language rdfs:label ?langlabel .
}
FILTER langMatches( lang(?langlabel), "nb" )
} GROUP BY ?record ORDER BY DESC(?issued) ?record
```

(It would probably have been better to re-write this query using DESCRIBE instead of SELECT, since that would have made it possible ti display multiple languages, authors etc for one document, but time did not allow for doing this.)

Detailed record display

Clicking on an entry in the list of results opens a pop-up dialog (courtesy of jQuery UI) that displays details about the record along with a cover image, if we have been able to find one.

Record detail

Details of one record are retrieved like this:

```
PREFIX bibo: <http://purl.org/ontology/bibo/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX pode: <http://www.bibpode.no/vocabulary#>
DESCRIBE <http://www.deich.folkebibl.no/cgi-bin/websok?tnr_0884778> ?
format ?creator ?editor ?publisher ?publicationPlace ?subject ?
language WHERE {
<http://www.deich.folkebibl.no/cgi-bin/websok?tnr_0904969>
dct:format ?format .
OPTIONAL { <http://www.deich.folkebibl.no/cgi-bin/websok?tnr_0884778>
dct:creator ?creator . }
OPTIONAL { <http://www.deich.folkebibl.no/cgi-bin/websok?tnr_0884778>
bibo:editor ?editor . }
OPTIONAL { <http://www.deich.folkebibl.no/cgi-bin/websok?tnr_0884778>
dct:publisher ?publisher . }
OPTIONAL { <http://www.deich.folkebibl.no/cgi-bin/websok?tnr_0884778>
pode:publicationPlace ?publicationPlace . }
OPTIONAL { <http://www.deich.folkebibl.no/cgi-bin/websok?tnr_0884778>
dct:subject ?subject . }
OPTIONAL { <http://www.deich.folkebibl.no/cgi-bin/websok?tnr_0884778>
dct:language ?language . }
}
```

Cover image

If the record detail contains an ISBN, an <img> tag is inserted with the src-attribute pointing to the image.php script, with the ISBN as an argument: image.php?isbn=<ISBN>.
image.php sends a request to the web service of Norwegian online book retailer bokkilden.no. If this request returns a response that contains the URL of a cover image for the ISBN in question, a redirect to that URL is sent to the browser, and the cover is displayed.
If no cover image was found, a redirect is made to a URL at the Open Library, on this form:
`http://covers.openlibrary.org/b/isbn/{$compact_isbn}-M.jpg`
If the Open Library has a cover image associated with the ISBN in question it is returned by this URL, otherwise a 1x1 pixle transparent GIF image is returned.

Installation

Prerequisites

- A web-server that supports PHP
- jQuery
- At least the “Dialog” component and a theme of your own choice, from the jQuery UI library, available from http://jqueryui.com/download

LinkedNonFictionWeb can be installed in two different ways. Pleas note: this will only give you a copy of the “web interface”, not the data held in the ARC triplestore. Queries directed at the triplestore should still work, since they are returned in JSONP format.

Installation from an archive

1. Download the source from https://github.com/pode/LinkedNonFictionWeb/ by clicking on the “Downloads”-button and choosing either the .zip or .tar.gz option. Unpack the files in a web-accessible directory on your server.
2. Create a directory called “jquery” inside the directory that was created by unpacking the archive above. Download at least the “Dialog” component, a theme and jQuery itself from http://jqueryui.com/download and place it in the jquery directory.

Installation with Git

If you have Git (http://git-scm.com/) installed, you can install LinkedNonFictionWeb with the following commands on the command line:

```
git clone git://github.com/pode/LinkedNonFictionWeb.git
cd LinkedNonFictionWeb
# Install jQuery
mkdir jquery
cd jquery
```

Download at least the “Dialog” component, a theme and jQuery itself from http://jqueryui.com/download unpack it and place it in the jquery directory.

To get any changes to LinkedNonFictionWeb, all you then have to do is move into the LinkedNonFictionWeb directory created above, and issue the following command:

```
git pull
```