# Pode - LinkedAuthors - 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. LinkedAuthors

## Dataset

The original dataset consisted of two separate exports from the catalogue of Deichmanske bibliotek, Oslo Public Library. These exports contain bibliographic MARC records expressed as XML. The bibliographic library record is mainly a description of a particular manifestation, but it also contains information that describes the resource more generally. The FRBR tool uses this information to construct instances for expressions and works, in addition to the manifestation instances that the catalogue originally describes, and to create relations between these instances. The output of the FRBR-ization process is also an XML document, where bibliographic data is still expressed in MARC fields:

FRBRized Petterson dataset as XML

Trond Aalberg, one of the creators of the FRBR-tool, has written a document that highlights the findings in the MARC-records and in the dataset. The document is in Norwegian. For a summary in English you can read the blogpost.

## Converting to RDF

The next step was to identify the different types of instances, relations and data described in the FRBR-ized dataset, and to map these to RDF classes and properties. This work resulted in the information model ontology for this dataset, as well as a set of directions for the converting. We mainly used contents from the Core FRBR and Dublin Core metadata terms vocabularies. However, we also found the need to create several new classes and properties in our own namespace, in order to express our data as concisely as we wanted.

The converting itself was done by an XSLT stylesheet, which took XML input and produced RDF data in Turtle syntax. The FRBR-ized XML dataset already containe unique identifiers for works, expressions and manifestations, as well as some persons. In the converting process we reused these, but also created RDF instances for the remaining persons, as well as publisher and places. To describe languages we used already existing instances in the Lexvo dataset.

The Petterson dataset in Turtle

Both the Turtle files and the information model were imported into and ARC triplestore, which provided us with a SPARQL endpoint for querying and retrieving the data.

# 2. LinkedAuthorsWeb

LinkedAuthorsWeb is a web-based display of the data created by LinkedAuthors, mashed up with data from a few other sources.

- Demo: http://bibpode.no/linkedauthors/
- Source code: http://bibpode.no/linkedauthors/

## Data sources

- LinkedAuthors - data stored in an ARC triplestore
  - SPARQL endpoint: http://bibpode.no/rdfstore/endpoint.php
- DBpedia
  - SPARQL endpoint: http://dbpedia.org/sparql
- Lexvo.org
  - Individual RDF files are available at e.g. http://www.lexvo.org/data/iso639-3/dan

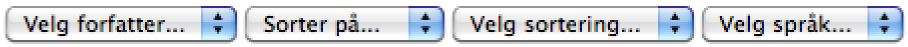
## Architectural outline

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

Initially, the user is presented with 4 drop-down form-controls:



These are, in order:
- Choose author (Knut Hamusn/Per Petterson)
- Sort by (publication year/title)
- Sort order (ascending/descending)
- Choose language (initially empty)

The main action taken by the user is choosing which author to display. This triggers 3 actions:
- A list of works by the chosen author is displayed on the left side of the screen.
- The drop-down of languages is populated.
- A box with information about the author is displayed on the right side of the screen.

### List of works

The list of works is requested as an AJAX-call against the LinkedAuthors ARC triplestore. The data are returned as JSONP, for easy handling of data. This is the query:

```
PREFIX pode: <http://www.bibpode.no/vocabulary#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX frbr: <http://purl.org/vocab/frbr/core#>
PREFIX person: <http://www.bibpode.no/person/>
SELECT DISTINCT ?work ?firstEdition ?title WHERE {
?work a frbr:Work ;
pode:firstEdition ?firstEdition ;
dct:title ?title ;
dct:creator person:Petterson_Per .
OPTIONAL { ?host frbr:part ?work } .
FILTER (!bound(?host))
} ORDER BY ASC (?firstEdition)
```

The query is integrated into a URL to retrieve the information:

```
http://bibpode.no/rdfstore/endpoint.php?query= + query + &jsonp=?
```

The query is modified if the user selects alternative values for what to sort by and the direction of the sorting.

## Language drop-down

When an author is selected, the language drop down is populated with all the languages that the current author is available in. The query for Knut Hamsun:

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX frbr: <http://purl.org/vocab/frbr/core#>
PREFIX person: <http://www.bibpode.no/person/>
SELECT DISTINCT ?language ?langlabel WHERE {
?work a frbr:Work ;
dct:title ?title ;
dct:creator person:Hamsun_Knut .
?expression frbr:realizationOf ?work .
?expression dct:language ?language .
?language rdfs:label ?langlabel .
FILTER langMatches( lang(?langlabel), "nb" )
} ORDER BY ?langlabel
```

If a language is chosen from the drop-down, the list of works is fetched again from the triplestore, limited to works that have expressions in the chosen language.

## Info-box

When an author is selected, the info-box on the right hand side of the screen is also populated with data from DBpedia. Several queries are used to do this:

1. A query is sent to the LinkedAuthors ARC triplestore, to get the DBpedia ID of the author:

```
PREFIX person: <http://www.bibpode.no/person/>
SELECT DISTINCT ?id WHERE {
person:Hamsun_Knut owl:sameAs ?id .
FILTER(REGEX(?id, "dbpedia"))
```

2. The DBpedia ID (URI) is used to query DBpedia. Since DBpedia does not serve JSONP, the request is made from proxy.php. The first request tries to collect all of the elements that have a single value:

```
SELECT DISTINCT * WHERE {
```

```
<http://dbpedia.org/resource/Per_Petterson> dbpedia-owl:thumbnail ?
thumbnail
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpprop:name ?
name } .
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpedia-
owl:activeYearsEndYear ?activeYearsEndYear } .
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpedia-
owl:activeYearsStartYear ?activeYearsStartYear } .
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpedia-
owl:birthDate ?birthDate } .
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpedia-
owl:birthName ?birthName } .
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpedia-
owl:birthPlace ?birthPlace } .
OPTIONAL { ?birthPlace dbpprop:name ?birthPlaceName } .
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpedia-
owl:deathDate ?deathDate } .
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpedia-
owl:deathPlace ?deathPlace } .
OPTIONAL { ?deathPlace dbpprop:name ?deathPlaceName } .
OPTIONAL { <http://dbpedia.org/resource/Per_Petterson> dbpedia-
owl:nationality ?nationality } .
OPTIONAL { ?nationality dbpedia-owl:thumbnail ?nationalityThumbnail }
 .
OPTIONAL { ?nationality dbpprop:nativeName ?nationalityLabel } .
}
```

3. An abstract describing the author is retrieved with this query:

```
SELECT DISTINCT * WHERE {
<http://dbpedia.org/resource/Per_Petterson> foaf:page ?page ;
dbpedia-owl:abstract ?abstract .
filter langMatches(lang(?abstract), "nn")
}
```

4. Authors influenced by this author are retrieved:

```
select distinct * where {
<http://dbpedia.org/resource/Per_Petterson> dbpprop:influenced ?
author .
?author rdfs:label ?authorLabel .
filter langMatches(lang(?authorLabel), "nn")
}
```

5. Authors this author has been influenced by are retrieved:

```
select distinct * where {
<http://dbpedia.org/resource/Per_Petterson> dbpprop:influences ?
author .
?author rdfs:label ?authorLabel .
filter langMatches(lang(?authorLabel), "nn")
}
```

## Expressions associated with a work

Clicking on a work in the list of works makes that work expand to reveal the Expressions that are associated with it. There are two similar queries, depending on whether a language has been chosen or not. The query when a language has been chosen:

```
PREFIX pode: <http://www.bibpode.no/vocabulary#>
PREFIX pode_lf: <http://www.bibpode.no/lf/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX frbr: <http://purl.org/vocab/frbr/core#>
PREFIX person: <http://www.bibpode.no/person/>
PREFIX work: <http://www.bibpode.no/work/>
SELECT DISTINCT * WHERE {
?expression a frbr:Expression .
?expression frbr:realizationOf <http://www.bibpode.no/work/
Hamsun_Knut_pan> .
?expression dct:format ?format .
?format rdfs:label ?formatlabel .
?expression dct:language <http://lexvo.org/id/iso639-3/nob> .
}
```

And when a language has not been chosen, main differences from the example above are highlighted in bold:

```
PREFIX pode: <http://www.bibpode.no/vocabulary#>
PREFIX pode_lf: <http://www.bibpode.no/lf/>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX frbr: <http://purl.org/vocab/frbr/core#>
PREFIX person: <http://www.bibpode.no/person/>
PREFIX work: <http://www.bibpode.no/work/>
SELECT DISTINCT * WHERE {
?expression a frbr:Expression .
?expression frbr:realizationOf <http://www.bibpode.no/work/
Hamsun_Knut_pan> .
?expression dct:format ?format .
?format rdfs:label ?formatlabel .
?expression dct:language ?language .
?language rdfs:label ?langlabel .
FILTER langMatches( lang(?langlabel), "nb" )
} ORDER BY ?langlabel
```

We also need to check if the work has parts (e.g. poems or short stories) that should be displayed:

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX frbr: <http://purl.org/vocab/frbr/core#>
PREFIX pode: <http://www.bibpode.no/vocabulary#>
SELECT DISTINCT ?title ?subtitle ?firstEdition WHERE {
<http://www.bibpode.no/work/Hamsun_Knut_siesta> frbr:part ?part .
?part dct:title ?title ;
pode:firstEdition ?firstEdition .
OPTIONAL { ?part dct:subtitle ?subtitle . }
}
```

When a new work is clicked, the previously chosen work is “collapsed”, hiding the list of parts and expressions. If a previously collapsed work is clicked again, it is “expanded” to reveal the hidden information, making it unnecessary to re-fetch the information from the server.

## Manifestations associated with an expression

When an expression is clicked, it is expanded to reveal it’s associated manifestations. The query:

```
PREFIX pode: <http://www.bibpode.no/vocabulary#>
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX frbr: <http://purl.org/vocab/frbr/core#>
SELECT DISTINCT ?title ?subtitle ?responsibility ?issued ?
physicalDescription WHERE {
?expression a frbr:Expression .
?expression frbr:realizationOf <http://www.bibpode.no/work/
Hamsun_Knut_sult> .
?expression dct:language <http://www.lingvoj.org/lang/nb> .
?expression dct:format <http://www.bibpode.no/ff/l> .
?expression frbr:embodiment ?manifestation .
?manifestation a frbr:Manifestation .
?manifestation dct:title ?title .
OPTIONAL{ ?manifestation pode:subtitle ?subtitle . }
OPTIONAL{ ?manifestation pode:responsibility ?responsibility . }
OPTIONAL{ ?manifestation pode:publicationPlace ?publicationPlace . }
OPTIONAL{ ?manifestation pode:publisher ?publisher . }
OPTIONAL{ ?manifestation dct:issued ?issued . }
OPTIONAL{ ?manifestation pode:physicalDescription ?
physicalDescription . }
}
ORDER BY ?title ?issued
```

## Installation

Prerequisites

- A web-server that supports PHP
- jQuery

LinkedAuthorsWeb 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/LinkedAuthorsWeb/ 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 jQuery from http://docs.jquery.com/Downloading_jQuery and place it in the jquery directory.

### Installation with Git

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

```
git clone git://github.com/pode/LinkedAuthorsWeb.git
cd LinkedAuthorsWeb
# Install jQuery
mkdir jquery
cd jquery
wget http://code.jquery.com/jquery-1.4.2.min.js
```

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

```
git pull
```