Metadata API and Management and Display of Bibliographies

Idea: *Bibliography management and display*

General goals:

- clean metadata with formal schemas and persistent identifiers
- metadata API in three parts: base REST API, content negotiation and export links
- interfaces for managing, editing and exporting publications

Tasks

Community bonding & planning

A number of things need to be discussed before going ahead with implementations. The context of these is mentioned below, in the tasks where the decisions are needed.

- a format for persistent identifiers
- a format for URIs for RDF/XML
- a format for URLs for the API
- data sources
- which API implementation to choose
- coordination is needed with developers working on 1) search results views and 2) journal display for bibliography exporting GUIs

A. Data cleaning

As the famous quote goes, data scientists spend between 60% to 90% of their time tidying data. There is a lot that can be done with the CDLI database:

- Persistent IDentifiers (PIDs): Currently for artifacts the database ID seems to correspond to the "CDLI No." from the old/current site (1 → P000001). I think it is important to preserve that somehow, as it would be *the* way to access the API. These identifiers should probably be registered for compact identifiers to create globally persistent identifers.
- Cleaning values: values have HTML entities, formatting, links that should be removed
- Extracting values: certain values like materials and dimensions are only available in plain text, not separate variables
- Merging data sources: if additional data sources are available, and they can be linked to the current data with provenance, they should be merged

B. Implement API

I propose a metadata API consisting of three parts. The main part (B1) is a versioned base API with a distinct subdomain or path that takes publication identifiers and formatting options as parameters. Formatting options may include output schemas, and pagination if necessary. The second part (B2) is a content negotiation API, similar to CrossCite, based on the regular URLs for artifacts/publications. The third part (C) is a GUI for exporting publications and artifacts.

Part of this task is to choose which formats to support. The main priorities are

- Internal format as XML/RDF and CSV/TSV
- BibTeX/BibLaTeX: similar format to internal format
- RIS (either edition)

If the schema of the internal is not yet documented, or does not have a URI, making sure it does will be as part of this step.

For the implementation details of external formats, I see three options (both behind a layer of CakePHP REST API for authentication):

1. Implement completely new format conversions (lot of work, simple tech, much control)
2. Deploy a Citation.js REST API (example) (little work, extra Docker container, I am the Citation.js author & maintainer)
3. Look for other libraries for conversions (not much control, possibly extra Docker container)

I would be interested to work together with other students to extends this API to journal management and display. Apart from querying metadata from CDLI's publications, it can be used to generate a BibTeX bibliography from the CDLI for LaTeX submissions. Additionally I can take up #126 too, since the tasks would be relatively related.

C. Update views for publications

The next step is to update views for publications to include links for exporting items:

- C1: for individual publications
- C2: for lists (browsing and search results) it will be possible to select individual publications. In case of search results, this should be coordinated with students working on regular and advanced search.

D. Implement publication data editor

Finally, views for adding, removing and editing publications will be implemented. These views will consist of forms containing all the data associated with a publications. Simple fields will be implemented first (D1), followed by autocomplete and search for linked fields like artifacts, authors and journals (D2).

Forms can be submitted in the form of a POST request to `/publication` or a PUT request to `/publication/<id>`. Additionally, these views should be able to handle CSV uploads for bulk editing, similar to QuickStatements for Wikidata. These would be POSTed to `/publication` as well. To allow updating records by CSV upload, POSTed entries with a specific, existing id will be treated as PUT requests to `/publication/<id>` (or vice versa).

In the case of editing, forms will be filled with current data. For user experience, a cancel button will be added to relieve any doubts that filling in the form somehow changes the database state; I know I tend to click cancel when possible.

Deleting publications, like exporting, will be possible by selecting items when browsing and in search results, and individual entries. The latter will be handled by DELETE requests.

Milestones

Progress will be tracked with GitLab issues and reports.

First evaluation (2020-06-29)

At this point, Tasks A and B should be done. This means that data is cleaned and a functional API is available, including content negotiation.

Second evaluation (2020-07-27)

At this point, Task C should be done. This means a tested and documented GUI to export bibliographies.

Final evaluation (2020-08-24)

At this point, Task D should be done. This means that a GUI editor for publication data is available, tested and documented.

Possible issues

- The data is very sparse at times, even compared to the seemingly structured data on the old/current site. This will possibly be fixed to some extent in the time allotted for data cleaning.
- Data cleaning, particularly the merging of data sources, is very dependent on communication with owners of said data sources.
- An editor with autocomplete features may be demanding for the database.
- I am new to PHP and CakePHP. However, I tried it out by making a view for single artifacts, which went well (!84).

Maintenance

For each task, time has been planned in to write tests and documentation for every part of code written where testing and such is applicable (C3, D3). Of course, code will be up to standard and be commented where necessary.

In addition to documentation and testing measures, I am also happy to help personally with maintaining the code in the long term, in terms of bug fixes and small improvements. Larger improvements and feature requests can be fine too, but that also depends on my schedule as a full-time Biology student. After GSoC, that will have to have some priority.

Timeline

Week	Monday	Sunday	Subtask	
19	2020-05-04	2020-05-10		Community bonding & planning
20	2020-05-11	2020-05-17		Community bonding & planning
21	2020-05-18	2020-05-24		Community bonding & planning
22	2020-05-25	2020-05-31		Community bonding & planning
23	2020-06-01	2020-06-07	A	Data cleaning
24	2020-06-08	2020-06-14	A	Data cleaning
25	2020-06-15	2020-06-21	B	Implement API (base API)
26	2020-06-22	2020-06-28	B1	Implement API (format conversions, see notes)
27	2020-06-29	2020-07-05		[First Evaluations]
28	2020-07-06	2020-07-12	C1	Update views for publications
29	2020-07-13	2020-07-19	C2	Update views for publications
30	2020-07-20	2020-07-26	C3	Testing, documenting
31	2020-07-27	2020-08-02		[Second Evaluations]
32	2020-08-03	2020-08-09	D1	Implement publication data editor
33	2020-08-10	2020-08-16	D2	Implement publication data editor
34	2020-08-17	2020-08-23	D3	Testing, documenting
35	2020-08-24	2020-08-30		[Students Submit Code and Evaluations]

Personal info

Hi, I'm Lars Willighagen and I am very passionate about bibliographical data. At age 15 I started programming a website to help classmates format bibliographies and since then I've been obsessed by anything metadata-related. This has led me past various projects and people related to bibliographical data, including Citation Style Language, DataCite, WikiCite and BibLaTeX. This summer, I hope to learn how to work from the side of a publisher and data owner, instead of the side of a data consumer.

- lars.willighagen@gmail.com
- site: https://larsgw.github.io
- GitHub: @larsgw
- GitLab: @larsgw
- ORCID: 0000-0002-4751-4637

Education & experience

- 2019 – : 1st year BSc Biology student at Radboud University, Nijmegen, Netherlands
- 2016 – : Citation.js author & maintainer
- 2019: 30th place (gold) in the 2019 International Biology Olympiad
- 2019: article *Citation.js: a format-independent, modular bibliography tool for the browser and command line* in PeerJ Computer Science
- 2018 – 2019: npm contributor and first forum community leader
- 2016: ContentMine fellow
- various open-source projects and contributions

Skills

- Extensive work parsing and mapping Bib(La)TeX into other formats
- Worked a lot with (modern) JavaScript & Node.js
- Worked with Rust, R and Python
- Worked with templating languages, HTML and CSS
- Contributing to and debugging existing open-source codebases (including npm, JASP, Wikidata)