Marriage of Figaro - Opera: Linked Open Data

Amanda Xu
Metadata Analyst
University of Iowa
amanda-xu@uiowa.edu

Abhishek Singh
Masters in Data Science
Indiana University, Bloomington
aasingh@umail.iu.edu

Anusha Ramamurthy
Masters in Data Science
Indiana University, Bloomington
anuramam@umail.iu.edu

ABSTRACT

With the rapid change in technology and connectivity, it has become necessary for libraries to have a revamp. A library is no longer just a catalogue of books, newspapers and other reading or viewing material. We wish to create a connected website for a library that leverages the technology available today, a linked open data online library.

Keywords

BIBFRAME 2.0; FUSEKI; SPARQL; RDF; LINKED OPEN DATA; JAVASCRIPT; BOOTSTRAP.js; HTML; HTTP ENDPOINTS;

1. INTRODUCTION

The Library linked data promises to meet libraries' need for agility in content delivery and user engagement. This project demonstrates the initial modeling of BIBFRAME 2.0 work, instance, item, agent, topic, etc. from local data examples in Opera Land, a collection of opera books, videos, sound recordings, streaming media, etc. interwoven into user's online experience using Alma, LC MARC to BIBFRAME Transformation Service, RDF Translator, RDF Validator, RDFa Validator, Apache Jena Fuseki Server, etc.

2. RELATED WORK

Our work is inspired by a previous project by Boston University Library, built on top of BIBFRAME 1.0. We tend to extend this further to BIBFRAME 2.0 and create an extensive catalogue.

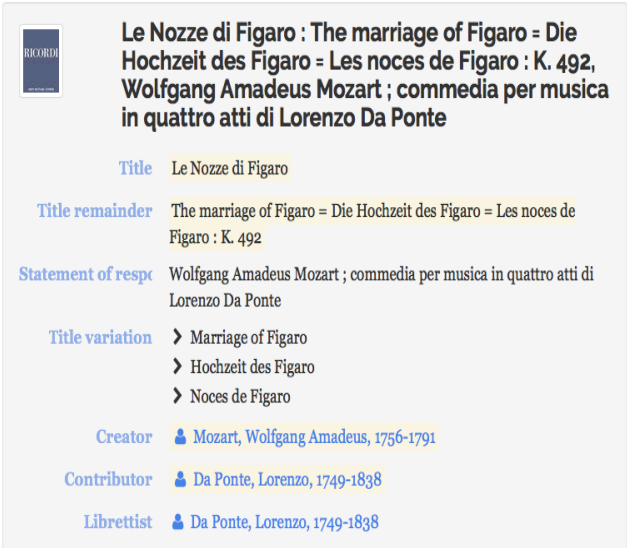

Figure [1]: Boston Library Page

3. ARCHITECTURE

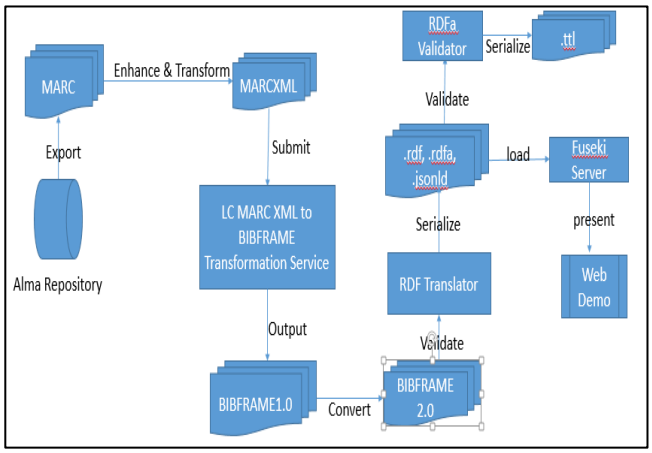

Figure [2]: Architecture

The architecture comprised of three main layers, the initial model for the data, the tools to extract rules and meaningful relationships from the data and the endpoint, to display the extracted content. For modeling our data, we used Alma Repository and exported it to MARC. The results from MARC tool were enhanced and transformed to MARCXML format. They were then submitted to LC MARQXML To BIBFRAME Transformation service layer. The output in BIBFRAME 1.0 had to undergo revisions to be updated to BIBFRAME 2.0 Model. It was then pushed through a validator to get formats like RDF, RDFa, Turtle and JSON-LD. The data in the RDF format was then stored on FUSEKI server running on localhost. SPARQL queries were used to extract rules and display data on the browser.

4. IMPLEMENTATION

The heading of a section should be in Times New Roman 12-point bold in all-capitals flush left with an additional 6-points of white space above the section head. Sections and subsequent sub-sections should be numbered and flush left. For a section head and a subsection head together (such as Section 3 and subsection 3.1), use no additional space above the subsection head.

4.1 BIBFRAME

BIBFRAME 2.0 organizes the information that provides the description of a resource into three core levels of abstraction: Work, Instance, and Item.

4.1.1 WORK

It is the highest level of abstraction. It reflects the conceptual essence of the cataloged resource: authors, languages, and what it is about (subjects).

4.1.2 INSTANCE

It is the resource reflecting an individual, material embodiment of a Work. It carries the info such as publisher, place and date of publication, and format.

4.1.3 ITEM

It is an actual copy (physical or electronic) of an instance. It bears the information such as location (physical or virtual), shelf mark, and barcode.

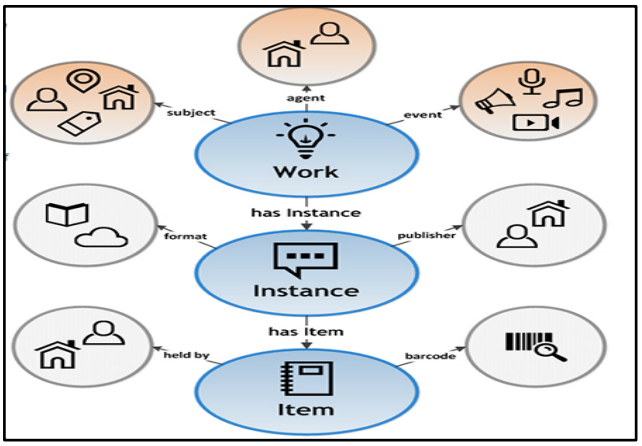

Figure [3]: BIBFRAME Model

4.1.4 Additional concepts related to BIBFRAME 2.0 core classes:

4.1.4.1 Agents: persons, families, organizations, jurisdictions, meetings, etc. associated with a Work or Instance through roles such as author, editor, composer, etc.

4.1.4.2 Subjects: A Work might convey one or more concepts that represent the "about" of the Work. Such a concept is referred as "subject" of the Work. Concepts may be subjects include topics, places, temporal expressions, events, works, instances, items and agents, etc.

4.1.4.3 Events: Details of what happens at a given place and time

4.2 FUSEKI

Our data was in several formats including rdf, which can be queried using SPARQL. Although this could be done on individual tools that return results to SPARQL queries, it was essential to our project that keep this process dynamic and send queries right from the website. This requires a server that can take these queries over HTTP and respond with the results accordingly. Fuseki was perfect for us, since it is a SPARQL server which provides REST-style SPARQL HTTP Update, SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP.

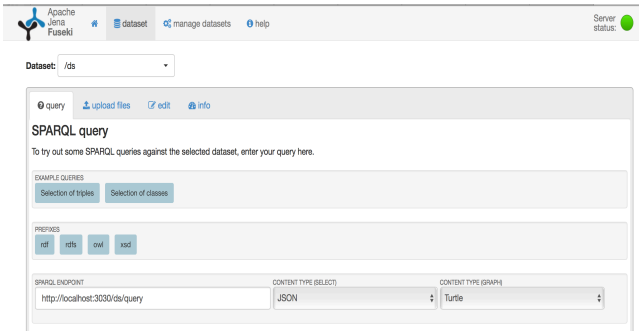

Figure [4]: Fuseki Server Interface

4.2.1 INSTALLATION

Fuseki server installation and setup is a comparatively easy process which can be summarized as follows:

First, we download and unzip the latest installation file (Apache Jena Fuseki 2.3.1) from jena.apache.org.

Launch terminal/cmd, change directory to apache fuseki folder and run 'fuseki-server' command. This will start fuseki server on localhost port 3030.

Open web browser and go to - **http://localhost:3030**, this will launch web interface for Fuseki Server.

Now, we can go ahead and setup the dataset by selecting a persistent dataset and importing our rdf/rdfa file into fuseki.

Once, the dataset is ready we can start querying. The results will be displayed right there in the browser interface.

4.2.2 SPARQL

Once we have Fuseki server up and running, we can write SPARQL queries to fetch exact information that we want from the RDF to display on our website. The formal Wikipedia definition of SPARQL goes as follows – 'SPARQL (pronounced "sparkle", a recursive acronym for SPARQL Protocol and RDF Query Language) is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format.'

We have written multiple SPARQL queries to fetch the data in particular format and granularity that we require. These SPARQL queries have been embedded in the webpage code itself.

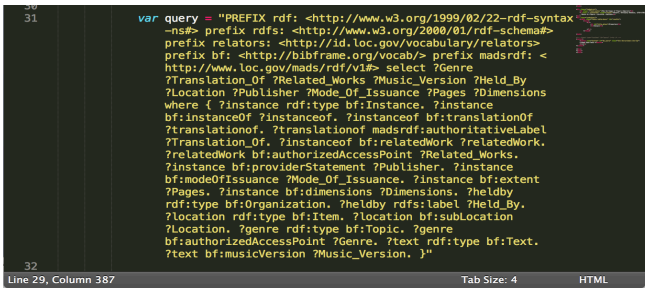

Figure [5]: SPARQL query used to build webpage

4.2.3 HTTP ENDPOINT

Fuseki server acts as our HTTP based SPARQL endpoint, since all our SPARQL queries have been pushed over HTTP to the Fuseki server that contains the tuples and serves them as and when requested over HTTP.

4.2.4 HTML-JAVASCRIPT

We have used JavaScript to make the page and queries dynamic. The JavaScript function is specifically designed to take the query results and transform them in a nice tabular format that looks both intuitive and beautiful on our webpage. We created javascript variable for the key-value pair and iterated over them to build our tables. Another noticeable thing here is that except query string, nothing is hardcoded, and the program can work well on any query result regardless of the format or size of result.

4.3 WEBPAGE

We are building a linked open data based library catalogue. Thus, it becomes necessary to demo a certain module. We created a website for demonstrating BIBFRAME catalogue for University of IOWA library. This website displays the information about an Opera – "THE MARRIAGE OF FIGARO". The information is

consistent with BIBFRAME models vocabulary- Work, Instance, & Item.

Figure [6]: Webpage

4.3.1 BOOTSTRAP

In order to build a website and demonstrate the linked open data functionality, we created a website using Bootstrap framework for a responsive user interface and JavaScript to query Fuseki server right from the webpage. Bootstrap is an open-source JavaScript framework developed by the team at Twitter. It is a combination of HTML, CSS, and JavaScript code designed to help build user interface components.

5. Roles

Amanda Xu: Data Curation. Creating the BIBFRAME Models

Anusha Ramamurthy: Fuseki Setup, Sparql Queries, Javascript

Abhishek Singh: Web page development, Bootstrap elements, javascript elements.

6. CONCLUSIONS

The project has been concluded with successful creation of a linked open data website for University of Iowa library catalogue with BIBFRAME 2.0.

7. FUTURE WORK

Current work is the first iteration of a large scale project. This can be further enhanced by - incorporating data about videos, vocal, sound recordings, streaming audio, score etc. related to the opera; bringing in more instances of publically available datasets like dbpedia, world cat etc; integrating with social networking websites, location services, Google custom search; PHP and asynchronous server side implementations; visualizations to see associations and related works.

8. ACKNOWLEDGMENTS

Our thanks to Prof. Ying Ding at Indiana University Bloomington for allowing us to collaborate on this project.

9. REFERENCES

[1] http://www.loc.gov/bibframe/pdf/marcld-report-11-21-2012.pdf

[2] http://link.bu.edu/portal/Le-Nozze-di-Figaro--The-marriage-of-Figaro--Die/UTfBiY8o1DY/

[3] http://jena.apache.org/documentation/fuseki2/

[4] http://www.w3schools.com/js/

[5] http://link.bu.edu/portal/Mozarts-The-marriage-of-Figaro-edited-by-Burton/EvMgjljaSlw/