Synia: Displaying data from Wikibases

Finn Årup Nielsen
Technical University of Denmark

Abstract

I present an agile method and a tool to display data from Wikidata and other Wikibase instances via SPARQL queries. The work-in-progress combines ideas from the Scholia Web application and the Listeria tool.

Keywords: Wikidata, Wikibase, SPARQL

Introduction

Scholia is a Web application running from the Wikimedia Foundation Toolforge server at http://scholia.toolforge.org. It displays data from Wikidata via SPARQL queries to the Wikidata Query Service (WDQS), particularly showing metadata about scientific publications (Nielsen et al., 2017), chemical information (Willighagen et al., 2018), and software (Rasberry and Mietchen, 2022). The Web application is implemented with the Python Flask framework and SPARQL templates are defined with Jinja2 templates that are read during the application startup and interpolated based on the Scholia user browsing. Two other tools use a similar Flask/SPARQL template approach to display Wikidata data: Ordia is specialized for the lexicographic part of Wikidata (Nielsen, 2019) and CVRminer[1] on Danish companies. Common limitations for these tools are currently

1. The tools are bound to the Wikidata WDQS endpoint

2. The language is fixed to English

3. Development of new panels and aspects requires the involvement of software developers.

For Magnus Manske's Listeria tool, wiki editors define MediaWiki templates with SPARQL queries on wikipages. The Listeria bot then edits on behalf of the user and generate tables on the wikipage according to the SPARQL query.[2]

The approach I will describe here was first explored in a specific instance of a Wikibase for data related to environmental impact assessment reports (Nielsen et al., 2023). In this abstract, I describe the extension of the approach, so that it can be used more widely with only slight changes in configurations in and across different Wikibases, — including Wikidata.

Methods

I call the tool *Synia* with the canonical homepage set up at https://synia.toolforge.org/. The implementation is a serverless single-page application (SPA) consisting of a simple HTML page and some JavaScript. Instead of storing the SPARQL templates along with the Web application, the templates are stored on wikipages. The URL pattern of Scholia is borrowed and changed to use URI fragments to control which wikipage should be read and what values should be interpolated in the template. Table 1 shows some of the mapping between the URI fragment and the wikipage. A pseudo-namespace, Wikidata:Synia, is used as the default for grouping the templates. If the template is not defined on the wiki, Synia creates a link, so a user/editor can create the template. Faceted search is supported, e.g., "#venue/Q15817015/topic/Q2013" shows information about the topic *Wikidata* occurring in the journal *Semantic Web*. Aspects with multiple items, e.g., handling "#authors/Q20980928,Q20895241,Q20895785" is not yet supported.

When wikipages are used for templates there are at least two important issues to consider: The template should be humanly readable as a wikipage and the information read should be untrusted as wikis are usually openly editable. Currently, a limited set of components are handled, see Table 2. The parsing of the components is based on a series of regular expressions. Synia will recognize MediaWiki headings and render them with h1, h2, and h3 HTML tags. SPARQL templates for Synia are stored on the wikipage in the *Template:SPARQL* MediaWiki template. Synia extracts the SPARQL code, interpolates the Q- and L- identifier(s), and sends the interpolated SPARQL to the SPARQL endpoint. The response is rendered as a table in the SPA using the DataTables JavaScript library or it may be rendered as a graph in an iframe with the graphing capabilities of the query service. For the ordinary wiki user, the template wikipage appears as ordinary wikipages with SPARQL as code examples,

---

[1] https://cvrminer.toolforge.org/.
[2] https://listeria.toolforge.org/.

see Figure 1. The wikipage may have multiple headings and SPARQL templates.

Other endpoints than the configured default can be queried. Currently Synia abuses an *endpoint* parameter for the *Template:SPARQL* MediaWiki template on Wikidata to specify the other endpoint. An example using the approach is currently displayed at `https://www.wikidata.org/wiki/Wikidata:Synia:compound` where a panel for a SPARQL query goes to the endpoint of the `https://wikifcd.wikibase.cloud` wiki (Thornton et al., 2021). This wiki has a Wikidata mapping property, so the Q-identifier can be matched across Wikibases to a Wikidata identifier.

Bootstrap, jQuery, and DataTables libraries are used. To avoid leaking browsing behavior the static files are hosted along with the SPA. Configuration, e.g., about the location of templates and the default endpoint is maintained in a separate JavaScript file.

A few aspects have so far been defined for Synia each with a few panels, e.g., author, work, venue, film, actor, compound, and lexeme. Figure 2 shows a screenshot of the actor aspect for the Wikidata entity Q294647 with two panels: a table and a bar chart.

To demonstrate that it is possible to use other template sites and other endpoints, I set up a template page at `https://www.wikidata.org/wiki/User:Fnielsen:Synia:index` copying a query from Wiki-FCD and reconfigured a cloned version of Synia to use "https://www.wikidata.org/wiki/User:Fnielsen:Synia:" as the template base URL and `https://wikifcd.wikibase.cloud/query` as the query service URL.

Discussion/Conclusions

The approach for the creation of new aspects and panels with Synia is more agile and wiki-like than Scholia's method. While the creation of a new panel in Scholia usually involves the creation of a new issue in GitHub, creation of a new branch, editing SPARQL and jinja2 code, commiting, pushing, merging the branch, testing, and deploying to Toolforge, a new panel with Synia is created by just editing a wikipage. Creating a new aspect with Synia can be done by creating a new wikipage, while for Scholia it would entail editing Python code as well as all the other steps involved in creating a panel. Discussions about new aspects or changes in Scholia take place on GitHub issue pages, while for Synia, discussions could take place on the wiki, e.g., the talk page associated with the templates.

Wikis with open editing, such as Wikidata, can be vandalized and security is an issue. If a malicious wiki editor adds a third-party endpoint then the browsing behavior of a Synia user will leak to the third-party site. The problem could be alleviated by having a set of allowed endpoints, e.g., Wikidata and Wikibase.cloud instances.

How language should best be handled is not clear. Figure 3 shows an aspect in Danish for a Danish company, so it is possible to control the language from a template. However, this approach "occupies" a specific URI pattern and a change of language is not possible without redoing much of the template.

Navigation with menu and search is currently missing in Synia as well as redirects and aspect-switching that all are available in Scholia. Instead of hardcoding such components in the Web application, it is envisioned that components in the templates on the wiki could control placement of menus and search forms.

SPARQL in MediaWiki templates may generate a problem as the pipe and the equality characters in SPARQL collide with the use of the characters to handle parameters in MediaWiki templates. Synia's simple regular expression parsing of the wikitext does not handle "{{!}}" that may be used to escape the pipe character in a MediaWiki template. A more elaborate parsing may be needed.

Acknowledgment

References

[Nielsen et al.2017] Finn Årup Nielsen, Daniel Mietchen, and Egon Willighagen. 2017. Scholia, Scientometrics and Wikidata. *The Semantic Web: ESWC 2017 Satellite Events*, pages 237–259, October.

[Nielsen et al.2023] Finn Årup Nielsen, Ivar Lyhne, Dario Garigliotti, Annika Butzbach, Emilia Ravn Boess, Katja Hose, and Lone Kørnøv. 2023. Environmental impact assessment reports in Wikidata and a Wikibase.

[Nielsen2019] Finn Årup Nielsen. 2019. Ordia: A Web application for Wikidata lexemes. *The Semantic Web: ESWC 2019 Satellite Events*, pages 141–146, May.

[Rasberry and Mietchen2022] Lane Rasberry and Daniel Mietchen. 2022. Scholia for Software. *Research Ideas and Outcomes*, 8, September.

[Thornton et al.2021] Katherine Thornton, Kenneth Seals-Nutt, and Mika Matsuzaki. 2021. Introducing WikiFCD: Many Food Composition Tables in a Single Knowledge Base. *Proceedings of the Joint Ontology Workshops 2021*.

[Willighagen et al.2018] Egon Willighagen, Denise Slenter, Daniel Mietchen, Chris T. Evelo, and Finn Årup Nielsen. 2018. Wikidata and Scholia as a hub linking chemical knowledge. *11th International Conference on Chemical Structures. Program & Abstracts*, page 146, May.
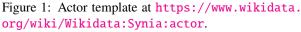
Template	URI fragment example
index	Main page
venue-index	#venue
author	#author/Q18618629
venue-topic	#venue/Q15817015/topic/Q2013
lexeme	#lexeme/L2310

Table 1: Mapping between URI fragments and wikipages with SPARQL templates.

Template	Handling
= Heading 1 =	h1 HTML tag
== Heading 2 ==	h2 HTML tag
=== Heading 3 ===	h3 HTML tag
----	hr HTML tag
{{SPARQL }}	Submitted to endpoint

Table 2: Handling of components on the wikipage.

Figure 1: Actor template at `https://www.wikidata.org/wiki/Wikidata:Synia:actor`.
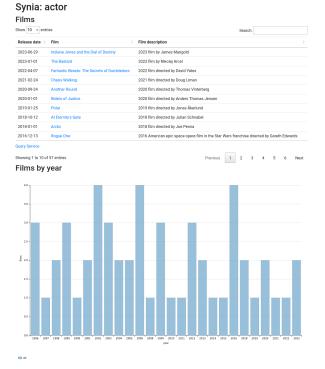

Figure 2: Rendered page for Wikidata entity Q294647 in the actor aspect of Synia at `https://synia.toolforge.org/#actor/Q294647`.
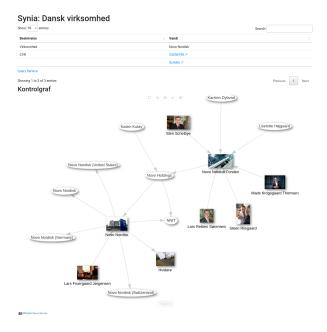

Figure 3: Rendered page in Danish for a Danish company, `https://synia.toolforge.org/#danskvirksomhed/Q818846`, with a control graph panel inspired from the CVRminer Web application.