SPARQL Views:
A Visual SPARQL Query Builder for Drupal

Lin Clark

Digital Enterprise Research Institute, National University of Ireland, Galway
`lin.clark@deri.org`

Abstract. Publishing Linked Data on the Web has become much easier with tools such as Drupal. However, consuming that data and presenting it in a meaningful way is still difficult for both Web developers and for Semantic Web practitioners. We demonstrate a module for Drupal which supports visual query building for SPARQL queries and enables meaningful displays of the query result.

Keywords: User Interfaces, End-user Programming, CMS

1 Introduction

Integrating Semantic Web technologies into mainstream Content Management Systems (CMSs) has been established as a way to increase adoption of the Semantic Web [3][2]. One example of incorporating these technologies in a mainstream system is the RDF in Drupal 7 core initiative, which enables over 300,000 sites[1] in joining the Semantic Web by exposing the content's structure as RDF.

While previous work in Drupal has made it easy for site administrators to *publish* RDF, modules that enable sites to *consume* RDF still require knowledge of SPARQL[2]. Because most Web developers do not meet this knowledge requirement, in practice the data is only directly accessible for Semantic Web practitioners. This means that one of the greatest potential benefits of RDF, the potential for data integration between sites, goes untapped for many sites.

Conversely, many Semantic Web practitioners do not have a full understanding of Web application development, which limits the displays they can create of the queried data. Where displays on the data are created, it is often by developing bespoke systems that cannot be reused by other Semantic Web practitioners.

We believe it is possible to design tools that support both the average Web developer and the Semantic Web practitioner's differing needs. We demonstrate a tool that supports drag-and-drop, visual query building over RDF data and that enables quick, easy, reusable presentation of that data.

[1] Drupal core usage statistics from `http://drupal.org/project/usage/drupal`

[2] Modules include SPARQL, http://drupal.org/project/sparql, and RDF SPARQL Proxy, http://drupal.org/project/rdfproxy

2 Use Case

For illustration of the tool and its usefulness, we imagine a research institute Web site such as `http://deri.ie/`. The site includes pages for researcher profiles. A researcher's profile page pulls the researcher's publication list from an dataset such as DBLP[3]. The profiles will be set up and administered by a Webmaster who is unfamiliar with SPARQL, but knows a small amount of PHP and is comfortable with HTML.

We walk through this use case in a video demonstration available at `http://lin-clark.com/iswc`. We give an analysis of the challenges faced in building such a Web site and our solutions to those challenges below.

3 Usability Challenges in Query Building

3.1 Figuring out where to start

Inexperienced users are overwhelmed when given too many interaction options or, even worse, when given a blank slate where they must enter a command language like SPARQL [5].

Interaction strategy—To assist users in finding where to start, we initially offer one single point of interaction. We make the assumption that most queries are centered around predicates, and we use the drag-and-drop predicate list as the first means of interaction. We then animate the addition of subject and object to that predicate in order to guide the user to the next step, using the strategy of sequential affordance[4].

3.2 Figuring out which predicates to use in the query

When accessing an arbitrary endpoint, end users have little support in understanding what data is contained in the dataset and how it is linked.

Interaction strategy—The endpoint is queried to determine which predicates are used in the dataset. Only predicates that are present in the dataset are displayed. An autocomplete search box allows the user to filter to appropriate terms.

3.3 Declaring prefix mappings

Declaring prefix mappings requires knowledge of both syntax and of standards and inexperienced users can have a hard time finding the appropriate namespace for a vocabulary.

Interaction strategy—The prefix declaration is automatically generated for the user from the predicates that are used. If a namespace mapping is not available in the system (which uses prefix.cc as it's source), then the full URI is used in the predicate display and the query.

[3] `http://dblp.l3s.de/d2r/sparql`

3.4 Understanding the logical structure of queries

While conjunction and disjunction based logic is ubiquitous in everyday life and is thus intuitive to a large degree, the syntax of SPARQL obscures that logic for the inexperienced user.

Interaction strategy—Where SPARQL syntax requires mental computation of the logic, visual representation of queries allows perceptual inference of the underlying logic[1]. We provide a visualization which makes the triple based logical patterns clear and offers affordances that reflect the possibilities of the graph structure. We currently only support basic conjunctive queries as these are the easiest to conceptualize[6], but are looking at ways to visually express other patterns.

4 Usability Challenges in Result Display

4.1 Displaying results in multiple display formats

SPARQL results are not useful in their raw format, but need to have tailored displays in order for the pattern of information within the data to be communicated. Semantic Web practitioners often create bespoke systems for this display, leading to duplication of effort within the Semantic Web community.

Interaction strategy—We integrated the query building tool with a widely deployed Drupal module, Views. This module offers a pluggable system for displaying query results. Switching from an HTML table view, to a Google API chart view, to a JavaScript Exhibit of the data is easy to do through the Views user interface. Because SV is integrated with this pluggable system, any style and display plugins that the Drupal community develops to display various kinds of data can also style SPARQL results.

4.2 Using context to rewrite the query

Information is most useful if it is tailored to the page where it is displayed. For instance, in the use case above, when a visitor visits a faculty member's profile, the query should return only publications authored by that faculty member.

Interaction strategy—We provide support for Drupal's Token API. This allows users to easily register tokens—small placeholder variables—and use these in their queries. The token is then evaluated at page load, so context, such as which user's profile is being viewed, can be assessed when creating the query.

5 Future Work

5.1 Predicate Preprocessing and Ranking

We currently use a DISTINCT query to get the predicates from a dataset. For certain endpoints, such as the one at `http://dbpedia.org/sparql`, this query will timeout and not return results. We plan to create a Web service that can extract predicates from crawls preprocess the predicates for different endpoints. This service could provide additional data, such as the predicate definition and a ranking based on predicate usage within the dataset.

5.2 Dataset Selection

A barrier we have not addressed in the current work is the selection of appropriate datasets. We plan to create tools to guide users in finding an appropriate dataset.

5.3 Federated Queries

Currently, there can only be one endpoint or dataset defined per view. However, one of the most exciting potentials of the Semantic Web is to mix data from multiple sources and we plan to explore how this can be supported in SV.

5.4 Evaluation

We believe that it is possible to support and encourage the user's increasing understanding of SPARQL by lowering the barrier to entry and focusing on learnability. We plan to evaluate this assertion, to see whether interaction with SPARQL Views increases the acceptance of SPARQL and the understanding of the syntax itself, enabling users to incrementally improve their understanding of the query language.

Acknowledgments. This work was funded by Google Summer of Code, LOD Around-The-Clock (LATC) ICT-256975, and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2). Thank you to Drupal contributors Daniel Wehner and Laura Scott for their guidance and feedback.

References

1. Tiziana Catarci, Maria F. Costabile, Stefano Levialdi, and Carlo Batini. Visual query systems for databases: A survey, 1995.
2. Stéphane Corlosquet, Renaud Delbru, Tim Clark, Axel Polleres, and Stefan Decker. Produce and consume linked data with drupal! In *The Semantic Web - ISWC 2009*, volume 5823 of *Lecture Notes in Computer Science*, pages 763–778. Springer Berlin / Heidelberg, 2009.
3. Markus Krötzsch, Denny Vrandečić, and Max Völkel. Semantic mediawiki. In *The Semantic Web - ISWC 2006*, volume 4273 of *Lecture Notes in Computer Science*, pages 935–942. Springer Berlin / Heidelberg, 2006.
4. Joanna Mcgrenere. Affordances: Clarifying and evolving a concept. In *Proceedings of Graphics Interface 2000*, pages 179–186, 2000.
5. Donald A. Norman. *The Design of Everyday Things*. Basic Books, New York, 2002.
6. Vladimir M. Sloutsky and Yevgeniya Goldvarg. Mental representation of logical connectives. *The Quarterly Journal of Experimental Psychology*, 57A(4):636–665, 2004.